



NTNU – Trondheim
Norwegian University of
Science and Technology

Developing a Web Application for Smart Home Technology

Øystein Waage
Jonas Lundeland

Master of Telematics - Communication Networks and Networked

Submission date: June 2012

Supervisor: Rolv Bræk, ITEM

Co-supervisor: Lars Kulseng, Trådløse Trondheim

Norwegian University of Science and Technology
Department of Telematics

Problem Description

By 2020, twenty percent of Europe's total energy reservoir will stem from renewable energy resources. In addition, the energy efficiency shall improve by twenty percent. These objectives are put forward by the European Union in what they call their 20-20-20 targets. In order to improve the energy efficiency in households, the consumers need to be made aware of their consumption and be offered ways of managing it. The Smarties project is a demonstration concept to show how energy efficiency can be improved through increased awareness. It is conducted by a group of Nordic companies consisting of Wireless Trondheim, Siemens, Kamstrup and several electricity suppliers. This thesis will focus on one of Wireless Trondheim's tasks, which is to develop the AMS web application. This application will run on the consumer's device and it will present data showing the consumption in both monetary and electrical units. It will also offer consumers the functionality of turning electrical devices on or off. By integrating these services with updated spot price information for the consumer's region, the goal is that economical motivation will lead to more energy efficient households.

This thesis will address the following issues:

Design and development

Produce an overall system design

Design an intuitive graphical user interface

Assess how electrical devices can be remotely controlled over the internet

Develop a working prototype of the web application

Present the users with tips on how to be more energy efficient

Execute performance tests and analyze the results

Security

Prevent unauthorized access to electrical devices

Assess different ways a system like this can be exploited by outsiders

Assignment given: 23. January 2012

Supervisor: Rolv Bræk, ITEM

Preface

The following work was conducted during the spring of 2012 at the Norwegian University of Science and Technology (NTNU) and it marks the completion of our Master's Degree in Telematics. The authors found the task of developing a web application for the Smarties project to be both exciting and challenging.

We would like to thank Lars Kulseng at Wireless Trondheim, for guidance and cooperation in the process of developing the AMS web application. We would also like to thank Rolv Bræk at the Department of Telematics for his guidance on writing this thesis. Finally we would like to thank our fellow students in *Futurum* and *Victoria* for contributing to a healthy work environment these past months.

Trondheim, June 7th, 2012

Jonas Lundeland

Øystein Waage

Abstract

With AMS comes great possibilities for increased energy efficiency, but to achieve its full potential, the end users must be provided with the necessary means of monitoring and controlling their consumption. This thesis describes the process of developing a web application prototype meant to serve such a purpose. It explains the various architectural and technological decisions that support the prototype, and it elaborates on how data from the users' smart meters can be synthesized with price information to help users see the economic effect of their current consumption pattern. A working prototype has been developed and security- and performance tests have been carried out to mitigate bottlenecks and prevent security breaches. Observations during the pilot project have shown promising trends and it is hoped that this thesis will inspire further innovation in the field of smart energy solutions.

Sammendrag

Introduksjonen av AMS legger til rette for økt energi-effektivitet, men for å få best mulig utbytte bør sluttbrukerne få tilgjengeliggjort nødvendig verktøy for å kunne overvåke og kontrollere forbruket. Denne avhandlingen beskriver hvordan en web applikasjon har blitt utarbeidet for dette formålet. Det forklares hvilke arkitektoniske og teknologiske avgjørelser som underbygger denne prototypen, i tillegg til å forklare hvordan data fra brukernes smarte målere kan kombineres med prisdata for å hjelpe brukerne med å se den økonomiske konsekvensen av forbruket sitt. En fungerende prototype har blitt utviklet, og sikkerhets- og ytelsestester har blitt utført for å begrense flaskehalsen og for å hindre sikkerhetsbrister. Observasjoner i løpet av pilotperioden har vist lovende tendenser og man håper at denne avhandlingen vil inspirere videre innovasjon innen smart energi-løsninger.

Table of Contents

Problem Description	i
Preface	iii
Abstract	v
Sammendrag	vii
Table of Contents	ix
List of Figures	xiii
List of Tables	xix
Acronyms and Abbreviations	xxi
Definitions	xxiii
1. Introduction	1
1.1. Limitations	2
1.2. The Smarties Project	2
1.3. System Overview	3
1.4. Web Application	8
1.5. AMS	9
1.6. The European Power Market	9
2. Methodology	13
2.1. Implementation	13
2.2. System Development	14
2.2.1. Specifying the Requirements	15
2.2.1.1 Developing High-level System Domain Models	15

Table of Contents

2.2.1.2	Developing Message Sequence Charts.....	17
2.2.1.3	GUI Models.....	18
3.	The Smarties system.....	27
3.1.	Similar Systems.....	27
3.1.1.	Google PowerMeter.....	27
3.1.2.	AlertMe.....	29
3.1.3.	TED – The Energy Detective.....	32
3.1.4.	Observation.....	34
3.2.	Deciding on Technologies.....	35
3.2.1.	Server-side.....	35
3.2.1.1	Java.....	36
3.2.1.2	Python.....	37
3.2.1.3	Deciding on a Framework.....	38
3.2.2.	Client-side.....	39
3.2.2.1	Using Purely Server-Side Logic.....	40
3.2.2.2	Why JavaScript?.....	40
3.2.2.3	Why jQuery Mobile?.....	41
3.2.2.4	Native Application.....	43
3.2.3.	In the Home.....	44
3.2.3.1	KNX.....	44
3.2.3.2	ZigBee.....	45
3.2.3.3	Z-Wave.....	46
3.2.3.4	Comparison.....	46
3.3.	Data Retrieval.....	48
3.3.1.	Meter Data.....	48
3.3.2.	Temperature and Price Data.....	51

3.4.	The AMS Web Application.....	53
3.4.1.	Presenting the Application	53
3.4.1.1	jQuery Mobile Page Scripting	53
3.4.1.2	Login Page.....	55
3.4.1.3	Start- and Pricelist Page.....	56
3.4.1.4	Control Page	58
3.4.1.5	History- and Datepicker Page	59
3.4.1.6	Settings Page	61
3.4.2.	Making the Data Available for the Client-Side	62
3.4.3.	Authentication	66
3.4.4.	Fetch Data from the Server.....	69
3.4.5.	Presenting Data and Drawing Graphs	71
3.4.6.	Controlling Devices in the Home	74
3.5.	Testing and Optimizing	77
3.5.1.	Identifying Bottlenecks	77
3.5.2.	Optimizing the Application	78
3.5.2.1	Compress the JS and CSS Files.....	78
3.5.2.2	Application Cache	79
3.5.2.3	Optimize the AJAX Requests.....	81
3.5.2.4	Session Data	82
3.5.3.	Performance Testing of the Web Application Prototype.....	82
4.	Security Assessment.....	87
4.1.	TR 1.6 Injection Attacks.....	89
4.2.	T-2 Cross-Site Scripting (XSS).....	92
4.3.	T-3 Broken Authentication and Session Management	96
4.4.	T-4 Insecure Direct Object References	97

Table of Contents

4.5.	T-5 Cross-Site Request Forgery (CSRF).....	101
4.6.	T-6 Path Traversal Attacks.....	103
4.7.	T-7 Insufficient Transport Layer Protection.....	105
5.	Discussion and Future Work.....	109
5.1.	Improvements to the Data-Collection.....	109
5.1.1.	Fetch Data from the Meter Logs.....	109
5.1.2.	Minimizing the KWh Collection.....	111
5.1.3.	More Tables in the Database.....	112
5.2.	Usage and Cost Comparison on the Start Page.....	113
5.3.	Upgrading the Pricelist Page.....	114
5.4.	Enhanced Power Saving Tips.....	115
5.5.	Control Devices in the Home.....	116
5.6.	SMS Notifications.....	119
5.7.	User-Customized Graphs.....	120
5.8.	Comparison with Similar Houses and Friends Feature.....	121
5.9.	Scaling the System.....	122
6.	Conclusion.....	123
	References.....	125
	Appendix A – The Smarties proposal template.....	133
	Appendix B – Initial requirements given by WT.....	147
	Appendix C – Email correspondence with AlertMe.....	149
	Appendix D – Appcache.....	151

List of Figures

Figure 1: The SMARTIES value chain as presented in the Smarties Proposal Template.....	3
Figure 2: The Smarties system overview	4
Figure 3: Spot prices differ between regions.....	11
Figure 4: Spot prices varies on an hourly basis.....	11
Figure 5: By introducing iterations between the various stages of system development a more flexible development process can be achieved	14
Figure 6: The System Context.....	16
Figure 7: MSC showing the interaction between the users' smart meters and the AMS server	17
Figure 8: MSC showing the user terminal authentication	17
Figure 9: MSC showing data retrieval from the AMS server.....	18
Figure 10: Device control is handled by the AMS server	18
Figure 11: GUI model of the Start page and the pricelist for the next 24 hours	19
Figure 12: GUI model of the History page.....	20
Figure 13: GUI models of the Control page.....	21
Figure 14: GUI model of the Settings page.....	22
Figure 15: Block diagram of the AMS Server showing the inner processes.....	24
Figure 16: The behavior of the price process	25
Figure 17: The behavior of the temperature process	25
Figure 18: Google PowerMeter allowed people to monitor their power consumption [15].....	28
Figure 19: An overview of AlertMe's system design [17].....	29
Figure 20: Detailed consumption information is presented in the AlertMe application [18].....	30
Figure 21: There is no need for a smart meter when using AlertMe [19]	31
Figure 22: The TED system uses several different ways of communicating in order to transmit usage data from the MTU to the user's devices [21]	33

Figure 23: TED’s wireless display presents the user with consumption information [20]	34
Figure 24: The web application part of the Smarties system	35
Figure 25: The Java platform [46].....	36
Figure 26: A Java servlet handling HTTP requests [24]	36
Figure 27: Separate tests showed that Pyramid performed better than Django for common web application tasks [32].....	39
Figure 28: The Smarties system could easily be changed to support native applications.....	43
Figure 29: Code snippet to make home screen bookmark on iOS devices	44
Figure 30: NATs represent the borderline between local and global address spaces	49
Figure 31: UDP hole-punching and first server-response	50
Figure 32: UDP ACK on initial HELLO.....	50
Figure 33: The meter reading sequence when consumption data is transferred to the server	50
Figure 34: TCP three-way handshake	51
Figure 35: TCP connection termination sequence.....	51
Figure 36: Each "page" is given a different id and this id is used for browsing between the different pages.....	54
Figure 37: Each page element from the index.pt file has a corresponding function in the dataFetchAndDraw.js.....	54
Figure 38: The Login page	55
Figure 39: The Start page	56
Figure 40: Prices for the next day is available in a separate page	57
Figure 41: Power saving tips are presented on the start page.....	57
Figure 42: The “Personal usage patterns” can help the user to discover unnecessary power usage	58
Figure 43: The Control page	59
Figure 44: The History page.....	60
Figure 45: The user can manually set the graphs’ time interval.....	60
Figure 46: The Settings page.....	61
Figure 47: Model showing the relations between different tables in the database	62

Figure 48: A view returning JSON data upon a request.....	63
Figure 49: The returned JSON data from the <i>/meterInfo</i> view.....	64
Figure 50: The JSON data returned from the <i>/initialData</i> view.....	65
Figure 51: A simple jQuery entry can disable the password manager for the given page	66
Figure 52: If the credentials are correct, the user gains access to the application	67
Figure 53: If incorrect credentials are passed to the server, access to the application is denied.....	67
Figure 54: The server-side authentication process	68
Figure 55: AJAX is used in <i>dataFetchAndDraw.js</i> to fetch JSON data from the server	69
Figure 56: The different AJAX requests in <i>dataFetchAndDraw.js</i> file.....	70
Figure 57: The function call for drawing graphs in the history page	72
Figure 58: A generic function for graph generation.....	72
Figure 59: Graphs showing various correlations.....	73
Figure 60: The resize function.....	73
Figure 61: An Arduino board was used to simulate device control functionality [53].....	74
Figure 62: A successful test of the home control proof-of-concept was performed	75
Figure 63: Circuit diagram showing the simulation setup.....	75
Figure 64: Database tables were created to facilitate device control functionality	76
Figure 65: The code in the HTML file that enables appcache	79
Figure 66: The difference in loading times with and without Appcache is significant.....	80
Figure 67: The graph data was stored in separate two-dimensional arrays.....	81
Figure 68: Temporarily storing user data as session data on the server helped reduce the server response times	82
Figure 69: The system can serve up to ten concurrent users without significant problems.....	83
Figure 70: Bottlenecks were discovered as the load was increased from ten to fifty virtual users	84

List of Figures

Figure 71: The ten most time-consuming responses were identified	85
Figure 72: After upgrading the hardware, decrease in loading times was observed	86
Figure 73: Response times after hardware upgrade.....	86
Figure 74: The user input fields on the authentication page were exploited in an attempt to perform an SQL injection attack	90
Figure 75: Server logs were used to monitor how the server reacted to an SQL injection attack	91
Figure 76: An excerpt from the server log after the second injection attack.....	91
Figure 77: By adding a client side script to a HTML link element, the malicious intent is hidden from the client.....	92
Figure 78: MSC showing the logic behind a reflected XSS attack	93
Figure 79: MSC showing the logic behind a stored XSS	94
Figure 80: The AMS application uses whitelist filtering when doing input validation. Inputs not present in the whitelist are ignored.....	95
Figure 81: User requests with the intention of changing a user's default meter ID were properly handled at the server side	99
Figure 82: By including a random token in the HTTP requests, CSRF attacks are counteracted.....	102
Figure 83: It was attempted to access a text file (outline 1) outside the application's root folder (outline 2)	103
Figure 84: Wireshark was used to capture unencrypted username and password	106
Figure 85: Wireshark was used to capture session ID.....	106
Figure 86: Tools such as Cookies Manager+ makes it easy for anyone to edit and add cookies.....	107
Figure 87: When SSL is enabled, the data is encrypted and is therefore incomprehensible to anyone monitoring the traffic.....	108
Figure 88: When connection is lost between the smart meter and the server, the data presented to the user is faulty for the down period.....	110
Figure 89: A suggestion on how the monthly comparison feature could look like	113
Figure 90: A suggestion on how to improve the pricelist feature	114
Figure 91: A timer functionality could help save electricity	117

List of Figures

Figure 92: A proposal for a user-customized graph functionality	120
Figure 93: A load balancer can help improve the system's scaling properties	122
Figure 94: Creating database clusters is a way of scaling the application to an increased amount of data.....	122
Figure 95: All the files in the manifest file are downloaded when there has been an update since last time the page was loaded.....	151
Figure 96: If no update has been done, the files are loaded from cache.....	151
Figure 97: Loading times without Appcache	152
Figure 98: Loading times with Appcache	152

List of Tables

Table 1: A dictionary was created and kept up to date throughout the system development process	16
Table 2: The various views that make data available to clients	64
Table 3: Unnecessary separation of data caused a bottleneck.....	78
Table 4: Assessing the different risks defined in OWASP list of most common attacks.....	88
Table 5: Characters often used in SQL injection attacks.....	91
Table 6: By creating a table with pre-calculated cost values, the loading time per request can be reduced	112

Acronyms and Abbreviations

AJAX	-	Asynchronous JavaScript and XML
AMS	-	Automatic Metering System ¹
API	-	Application Programming Interface
CPU	-	Central Processing Unit
CSRF	-	Cross Site Request Forgery
CSS	-	Cascading Style Sheets
DoS	-	Denial of Service
GUI	-	Graphic User Interface
HAN	-	Home Area Network
HTML	-	HyperText Markup Language
HTTP	-	HyperText Transfer Protocol
IE	-	Internet Explorer
ISO	-	International Organization for Standardization
IP	-	Internet Protocol
JSON	-	JavaScript Object Notation
JS	-	JavaScript
KWh	-	Kilowatt hour
MSC	-	Message Sequence Chart
MTU	-	Measuring Transmitting Unit
NAT	-	Network Address Translation
NOK	-	Norwegian Kroner
OS	-	Operating System
OWASP	-	Open Web Application Security Project
PLC	-	Power Line Communication
RFC	-	Request for Comments
SHA	-	Secure Hash Algorithm
SSL/TLS	-	Secure Sockets Layer/Transport Layer Security
SQL	-	Structured Query Language
TCP	-	Transmission Control Protocol

¹ In Norwegian: Avanserte målings- og styringssystemer

Acronyms And Abbreviations

TED	-	The Energy Detective
UDP	-	User Datagram Protocol
URL	-	Uniform Resource Locator
WAP	-	Wireless Access Point
WT	-	Wireless Trondheim
XSS	-	Cross Site Scripting

Definitions

All definitions are gathered from TechTerms.com [1].

AJAX – Combines JavaScript and data-interchange formats such as JSON and XML for asynchronous data collection. Using it for data collection, results in more dynamic web sites.

API – Application Program Interface is a collection of commands, methods and protocols, necessary for developers when creating software intended to support or make use of existing systems.

CSS – Cascading Style Sheet for defining the layout of HTML elements.

GUI – Graphical User Interface refers to the graphical appearance of the application. The use of graphical elements such as buttons makes for a more user friendly interface, compared to command line.

JSON – JavaScript Object Notation is a text based data-interchange format designed for fast and asynchronous data transfer. As the name implies, it is easy to implement in JS.

SSL – Secure Socket Layer is a transport layer protocol for sending information securely over the internet. By encrypting the data before transmitting it, SSL protects the data against eavesdropping.

1. Introduction

As one of several measures to reduce greenhouse gas emissions, the European Commission approved in December 2008 the so-called *renewable energy directive* [2]. This directive seeks to dramatically increase the energy efficiency of European households. The environmental benefits put aside, there are still other reasons for being energy efficient, namely to save money. However, without being presented with conclusive facts and tips on how to save electricity, people are seemingly not motivated to do something about it. It is therefore clear that there is a need for a system providing such a service.

After completing their pilot project, CenterPoint Energy and U.S. Deputy Secretary of Energy, Daniel Poneman released a survey [3] showing that by offering end users ways of monitoring their consumption, remarkable improvements in power efficiency can be observed. According to the same survey, a total of seventy-one percent of the questioned acknowledged a change in their consumption pattern as a consequence of being presented with detailed energy usage data.

Based on the idea that increased awareness makes for more energy efficient end users, a group of Nordic companies got together and formed a pilot project group in 2011. The group would explore the possibilities of utilizing already existing infrastructure to retrieve consumption data and present this data to the user in a web application customized for touch devices.

The purpose of this thesis is to elaborate on the process of developing a working prototype for the web application. The reader will get insight into the various design- and technology-related decisions, as well as a review of the finalized prototype.

1.1. Limitations

Although the different issues mentioned in the problem description have been addressed in this research, the authors are aware of some limitations. There was an uncertainty, throughout the project, regarding who would provide the home control equipment necessary for controlling electrical devices in the users' household. The decision remained unsettled until late May, when it was decided that Bravida would replace Siemens in the project's value chain. Consequently, this led to uncertainties regarding which home control protocol was to be used, as well as not having access to the necessary hardware to fulfill the home control functionality. As a workaround, the authors acquired open-source equipment and simulated a home control environment.

The pilot period started in late April, and has not been active long enough for feedback to be provided from the pilot customers. This means that all the features and GUI designs are yet to be settled. An additional challenge was the relatively short deadline for finalizing a working prototype. As shown in the Gantt chart presented in Appendix A – *The Smarties proposal template*, the was to be finished by the end of April, leaving only two months for development. This meant that there was not enough time to implement every idea, but the features that were left out will be presented as suggestions to future work in chapter 5.

1.2. The Smarties Project

The SMARTIES workgroup is a collection of Scandinavian companies. According to the proposal template given in Appendix A, the group is to come up with a demonstration concept for which they have set forth three goals:

- Present how new technology can help reduce energy consumption
- Test new AMS technology on existing infrastructure and potentially reduce the cost of AMS by 30-50 percent
- Show how the consumer and the society can benefit from the massive rollout of AMS planned throughout Europe by 2020

The five workgroup participants all have their respective areas of responsibilities. The original value chain is illustrated in the Figure 1:

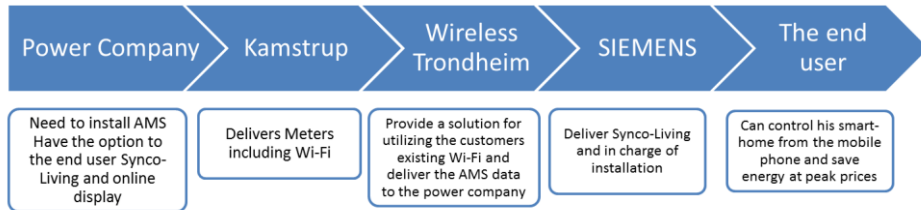


Figure 1: The SMARTIES value chain as presented in the Smarties Proposal Template

AMS systems currently available are mainly using Power Line Communication (PLC) and GSM/3G as the transport medium. The SMARTIES workgroup believe that using Wi-Fi as an alternate technology will offer benefits in addition to the aforementioned 30-50 percent cost reduction: *“GSM/3G and PLC are limited regarding channel capacity. By making use of Wi-Fi in the customer’s household and using internet as the transport medium the SMARTIES workgroup believe that the system will scale better as the user base grows.”*

During the demonstration project, the workgroup will monitor the end users’ energy consumption in order to detect changes in consumption patterns. The results will be presented in a report which the Smarties group hopes will act as a source of motivation to consumers. By showing how easy it is to save both energy and money by increased awareness and by making small changes to how energy is used, it is hoped that the consumers will see the benefit from acquiring such a system.

1.3. System Overview

WT’s role in the Smarties project encompasses the collection, processing, and presentation of power consumption data, making it possible for providing functionality for controlling the home over the internet. In order to get a good understanding of how the system is composed, a sketch showing the system overview of the entire system is presented in Figure 2. More detailed

information on how the system has been realized will be presented in section 2.2.

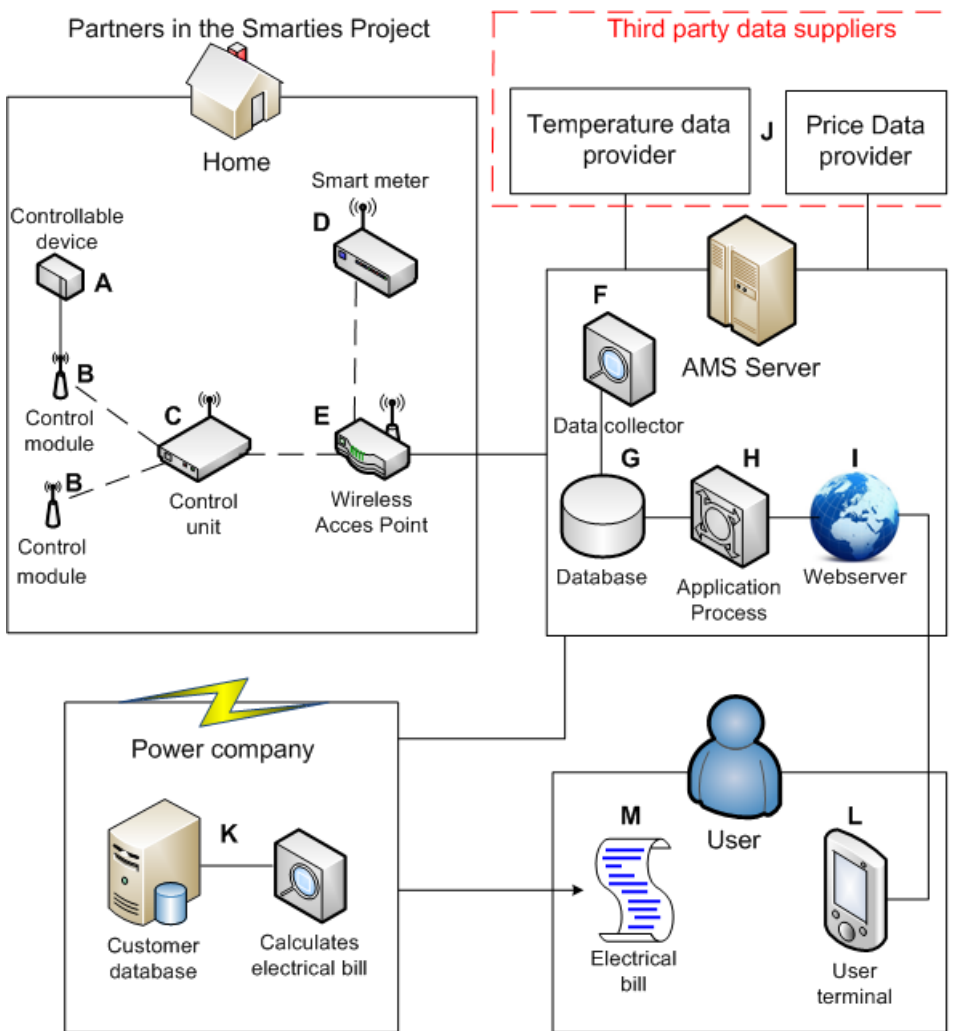


Figure 2: The Smarties system overview

Figure 2 shows the main areas of the system, where the different system entities marked from A to K. These entities are introduced below:

- The user's home
Several in-home devices are needed for the Smarties system to function as intended. A precondition for this design is that the user has a wireless access point (WAP) through which the different devices connect to the internet. It would be possible to use cabled Ethernet instead, but the Smarties group wants to explore how this can be achieved with as little installation effort as possible, and therefore Wi-Fi is the chosen way of communication.
 - A **Controllable devices** can be any electrical device in the home that can be turned on/off. In the Smarties project, the focus will be on power-consuming appliances like the water heater, under-floor heating and electric heaters.
 - B To be able to control appliances in the house, **control modules** are necessary equipment. By adding a control module between the appliance and the power source, the control module can start/stop the electricity flow, hence turning the appliance on/off on demand.
 - C The **control unit** is the in-home device that interacts with the AMS server. It receives control signals originating from the authorized user terminal, and the control signal is forwarded to the correct control module in order to execute the command.
 - D **Smart meters** [4] are very similar to normal electrical power meters, but they differ in the way they automatically registers the power consumption with intervals of no more than one hour. These results are normally transmitted back to the power supplier in order to give the user a much more correct electrical bill. The smart meters delivered by Kamstrup registers consumption data every five minutes and sends the data to the AMS server.
 - E The **WAP** is a common device to have in every home connected to the internet. This allows for wireless smart meters and control

units to easily communicate with the outside world with as little installation efforts as possible.

- The AMS server

The server setup of the Smarties project can be done in several ways. Figure 2 shows the setup for this pilot-project, with all the different services running on a single server. If the system gains popularity and experiences an increased customer base, it may be required to physically separate the services. Another possible solution is to outsource the database-part to the power companies. This will provide load-sharing on different servers and clustering of user data according to their associated power company. These measures combined would have a positive effect on the system performance.

F At this point there are three different processes running on the server that collects

- power consumption data from the smart meters
- temperature data
- hour-by-hour price information

G The **database** is where all the information collected from the different users' smart meters is stored. Price- and temperature data fetched from third-party suppliers is also stored in the same database. This this data is then utilized by the webserver to present relevant information to the user, and by the electric company to calculate the electrical bill.

H The **application process** is the actual application running on the server, which means that this is where all the logic behind the application is implemented. The process runs the code that fetches the appropriate data from the database and forwards it to the webserver so it can be displayed in the user's browser.

I The webserver acts as the AMS server's interface towards the users and it is accessible over the internet on a specific IP address. The web server will receive requests from the users and forward these requests to the correct application process. It will also be the exit point for responses going the opposite way, thus

being the optimal place for implementing measures for improved transport-layer security.

- ⌋ The collection of both **price** and **temperature** data is crucial in order to provide valuable information and advice to the user. Price data is used to calculate how much the user's power consumption is costing at any given time. The correlation between temperature-, usage-, cost- and price data can expose patterns showing how the individual user can save electricity as well as money.
- The power company
 - Ⓚ The design of the Smarties system depicts that the power company will retrieve usage data for its customers from the database described in G. Based on this data the electrical bill is calculated. This step will not be realized in this trial project because of some discrepancies with the service providers. However that has no influence on how the rest of the Smarties system is to be designed, thus not resulting in any problems at this point.
- By making consumption data available to the **user**, it is much easier for the user to increase the efficiency of his power consumption, and hence save both the environment and money
 - Ⓛ For this system to be of any value to the user, it is important that the application is easily accessible. Therefore it has been designed to function on both mobile devices like smart phones and tablets, as well as computers using regular web browsers. By using any of these devices this application can be opened from wherever in the world through the internet.
 - Ⓜ By utilizing this system with the smart meter, the user will receive a more correct **electrical bill** where the cost depends on when the electricity is used.

1.4. Web Application

Since the rise of web 2.0², there has been an increase in the number of websites that fit the term *Web Application*. Web applications emulate desktop applications, thus differing from the classic text and image based documents that make up the majority of the web [5]. Examples are online word processors such as Google Docs, Photo editing tools such as Picasa and mapping sites such as Google maps. With the introduction of HTML5, developers now have new available functionality at their hands. Features such as video tags, drawing and drag-and-drop capabilities all contribute to fancier and more advanced web applications.

The layout of a web application is defined in HTML and CSS documents, while the logic is implemented at the server-side and/or by using browser-supported languages, such as JavaScript on the client side. When an HTML document is loaded, the accompanying JavaScript and CSS files are also retrieved. For the purpose of streamlining the user experience, these files can be loaded once on the initial page request. For subsequent requests, AJAX provides dynamic client-server data interchange, allowing the client script to contact the server for storing/retrieving data without downloading entire web pages. This enhances the user experience as it reduces the site's loading time.

It is also possible to configure a web application to work even when the internet connection is lost. Such applications are aptly named *offline web applications*. The fact that a web application works in an offline state might sound like a contradiction to some, but by downloading and storing HTML, CSS and JavaScript locally when in an online mode, these files are accessible from cache memory once connection is lost.

² Web 2.0 is not a reference to a technical specification update, but rather it is referring to the cumulative changes in web developers' and end-users' way of using the web with more user-centered and interactive web pages.

1.5.AMS

As global energy consumption is on the rise and people become increasingly aware of the dangers of climate changes, more people seek to reduce their own energy usage. Many of these motivated consumers assess the electrical components' energy efficiency when acquiring new home appliances. Measures like this help reduce energy consumption, but greater benefits are achievable through conservation and awareness. AMS facilitates such improvements.

AMS is a generic term for both hardware and software in smart metering systems. Because of the possibility of automatic and frequent collection of meter readings, the accuracy of the power usage information is enhanced. In addition, one eliminates the efforts required by the end user to manually report his power usage, which is usually done every three months [6]. With the introduction of spot prices in Europe, where every hour of the day is priced separately, updated usage information will also result in a more accurate power bill.

The features of AMS can be incorporated in systems offering users ways of monitoring their power consumption. By combining consumption data with updated power prices, these users will also see the economic effects of the energy consumption. It is believed that they will consider measures to avoid unnecessary energy usage, which in turn will result in positive climate effects.

1.6.The European Power Market

In 1991, Norway adopted the energy act [7], aiming at reforming the production, transformation, transmission, distribution and consumption of energy. This act deregulated the power market, paving the way for a free market for buying and selling electrical energy and hence securing a market-driven energy development, less prone to negative effects of forecasts and political decisions. The energy act imposes on the local energy production companies to deliver energy, while at the same time allowing the consumers to choose from a variety of power companies. By adopting the energy act, Norway became a groundbreaker with respect to liberalization of the power market [8]. In 1993

NASDAQ OMX Commodities Europe³ were established. This further revolutionized the power market with its auction trade system for power contracts. On January 1st 1996, Sweden joined the market and the first multinational exchange for electrical power trading was established [9].

The Scandinavian initiative showed that available power capacity can be used more efficiently in large regions compared to smaller ones and it inspired several other European countries into establishing similar regional markets. The goal of the European Market Coupling Company (EMCC) is to couple all regional markets into one huge pan-European market.

The leading power market in Europe today is run by Nord Pool Spot, a company partly owned by electricity transmission system operators from the different Nordic countries [10]. According to Nord Pool's website, as much as 74 percent of all energy production in the Nordic region is traded on the Nord Pool market and in 2010 the total volume traded was 310 TWh, corresponding to a value of EUR 18 billion⁴.

Nord Pool Spot, as the name implies, offers spot prices where the price varies between different locations as illustrated in Figure 3.

³ Former Nord Pool ASA (then Statnett Marked AS)

⁴ Auction volume on N2EX which is the market Nord Pool and NASDAQ OMX Commodities operate in UK [59] is included.

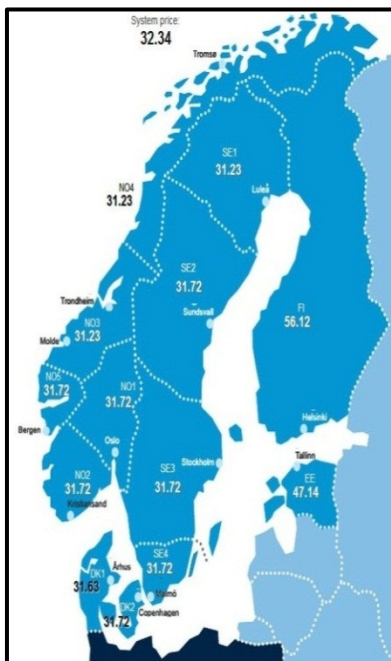


Figure 3: Spot prices differ between regions

	Oslo	Kr.sand	Bergen	Tr.heim
07-03-2012				
00 - 01	30,28	30,28	30,28	30,28
01 - 02	30,23	30,23	30,23	30,23
02 - 03	30,01	30,01	30,01	30,01
03 - 04	30,16	30,16	30,16	30,16
04 - 05	30,28	30,28	30,28	30,28
05 - 06	30,33	30,33	30,33	30,33
06 - 07	31,82	31,82	31,82	30,79
07 - 08	33,35	33,35	33,35	31,48
08 - 09	34,89	34,89	34,89	32,14
09 - 10	33,02	33,02	33,02	32,48
10 - 11	32,75	32,75	32,75	32,41
11 - 12	32,40	32,40	32,40	32,25
12 - 13	32,21	32,21	32,21	32,09
13 - 14	32,09	32,09	32,09	31,72
14 - 15	31,91	31,91	31,91	31,66
15 - 16	31,72	31,72	31,72	31,19
16 - 17	31,74	31,74	31,74	31,20
17 - 18	32,18	32,18	32,18	31,46
18 - 19	32,50	32,50	32,50	31,60
19 - 20	32,41	32,41	32,41	31,51
20 - 21	32,10	32,10	32,10	31,51
21 - 22	31,70	31,70	31,70	31,70
22 - 23	30,88	30,88	30,88	30,88
23 - 00	30,21	30,21	30,21	30,21
Min	30,01	30,01	30,01	30,01
Max	34,89	34,89	34,89	32,48
Average	31,72	31,72	31,72	31,23
Peak	32,49	32,49	32,49	31,81
Off-peak 1	30,81	30,81	30,81	30,45
Off-peak 2	31,22	31,22	31,22	31,08

Figure 4: Spot prices varies on an hourly basis

In addition, the price within each region varies on an hourly basis, shown in Figure 4. As soon as AMS is introduced and the power companies receive updated power usage information from the end users, an hourly pricing system can emerge. As the end users are presented with the price information of the following 24 hours, they can plan their energy consumption in order to reduce the cost.

2. Methodology

In order to secure a constructive progress in any project, it is important to settle on a methodical and suitable work plan at an early stage. This chapter will present the work process used in this project.

2.1. Implementation

One of the very first tasks in the project was to formulate a solid problem description. WT has a substantial area of responsibility in the Smarties project, it was therefore important to limit the scope of the assignment in order to keep focus on the most important tasks.

After limiting the scope, a progress plan was developed. This had to be coordinated with the pre-existing progress plan for the Smarties project presented in Appendix A – *The Smarties proposal template* in order to ensure that all goals were reached within the given time limits. The progress plan was continuously revised and updated in cases where tasks were accomplished on less or more time than estimated, which was necessary in order to keep a continuous status overview of the progress. In the following section, the work method used in the system development process of the AMS web application will be explained. After the commissioning of the application, performance and security tests were executed in addition to a critical assessment followed by suggestions for further development of the system.

2.2. System Development

Throughout this project the adopted methodology has been based on guidelines put forward by Bræk and Haugen in their compendium *Engineering Real Time Systems – An Object Oriented Methodology using SDL* [11]. The methodology presented here addresses the importance of solid planning and design prior to the system implementation. Both cost and labor hours can be reduced when efforts are put into the development of well thought-out system design models. Another important argument for selecting this methodology is its unambiguous trait, important when communicating system designs. It is vital that all parties involved in a system development process share the same understanding in order to avoid unnecessary design flaws. Since this system is developed as a collaborative project between several companies, the need for clear and unambiguous models seemed evident.

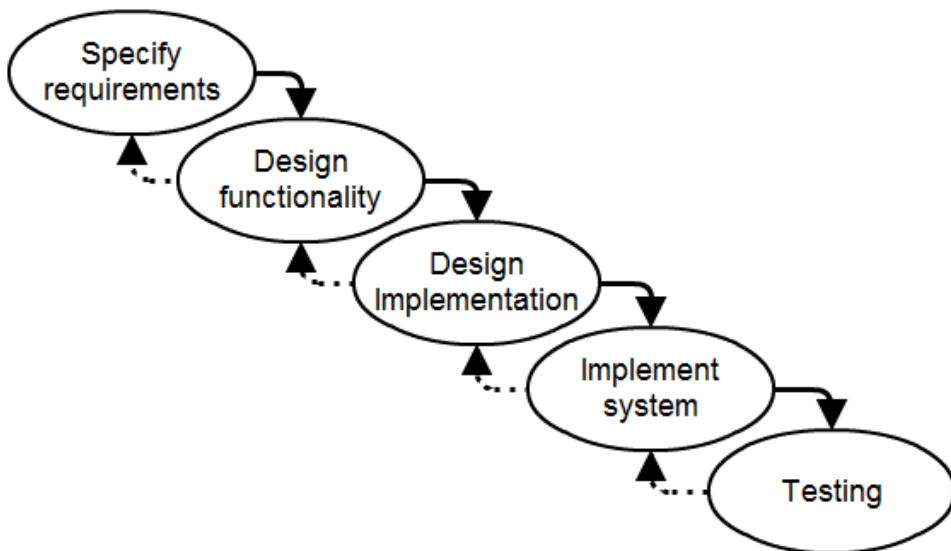


Figure 5: By introducing iterations between the various stages of system development a more flexible development process can be achieved

Even though the system development process illustrated in Figure 5 looks quite similar to the waterfall model, there is a significant difference, namely the iterations presented by the dotted arrows. Bræk and Haugen note the importance of feedbacks and iterations between the stages of system development. Should

something unforeseen happen during the development process, the iterations would allow the developer to take one step back in order to fix the issue. Such events could be:

- The system requirements changes during an ongoing development
- Developer gains greater understanding of how the system works
- Defects are discovered
- Flaws in design are discovered
- Optimization of already functional code

2.2.1. Specifying the Requirements

The system requirements were initially developed by Wireless Trondheim and this draft can be found in Appendix B – Initial requirements given by WT. After meeting with WT, some of these requirements were audited and new ones were added. The resulting requirement specification was then modeled. Different documents were produced during this phase:

1. **High-level system domain models.** Useful in the process of establishing a common understanding of the system domain
2. **Message Sequence Charts (MSCs).** Intuitive charts used to document interface behavior
3. **GUI mock-ups.** GUI sketches, useful when discussing GUI alternatives as well as templates for the programmers when developing the application

2.2.1.1 Developing High-level System Domain Models

At an early stage of the project, the system documentation was limited, which meant that models had to be developed on the basis of oral presentations given by representatives from Wireless Trondheim. The different system entities were identified and their relations were outlined in the models. This work resulted in the system context model, shown in Figure 6.

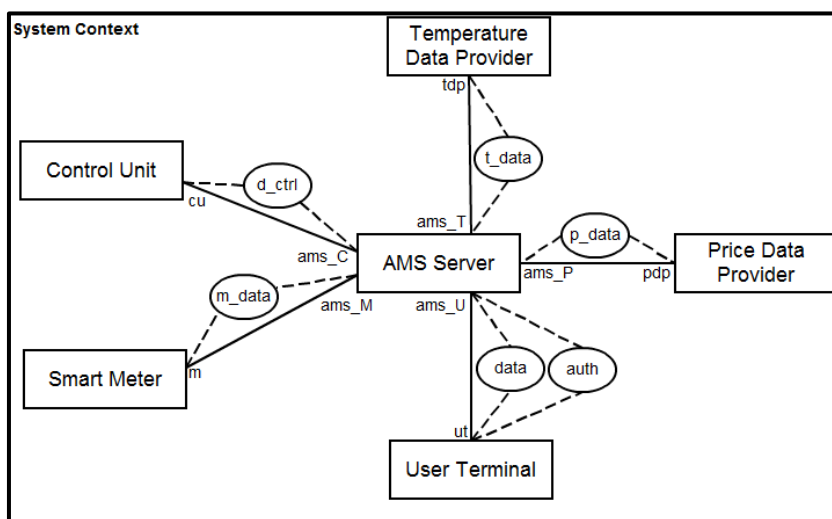


Figure 6: The System Context

The system context model served as an intuitive depiction of the problem domain and was invaluable to ensure common understanding between WT and the authors. To avoid ambiguity, a dictionary was developed for the problem domain. This dictionary served as a precise definition of the various concepts and terminologies applied in the system context model.

Table 1: A dictionary was created and kept up to date throughout the system development process

Dictionary for problem domain	
User Terminal	An AMS customer’s terminal. Smart phone or computer
Smart Meter	Kamstrup Smart Meter. Sends periodically meter readings to the AMS server
Temperature Data Provider	Meteorological Institute offering continuous temperature data
Price Data Provider	Nord Pool Spot. Offering hourly price information
Control Unit	Unit controlling electrical device(s) in the AMS customer’s household

2.2.1.2 Developing Message Sequence Charts

By depicting typical interaction sequences in MSCs, one can better communicate the interface logic. This documentation can be a visual aid for understanding the role each of the system entities plays in the system context. For this reason, there was developed MSCs for each of the collaborations in Figure 6. At this stage, the implementation details were still undetermined, hence the creation of high-level MSCs with references to underlying MSCs.

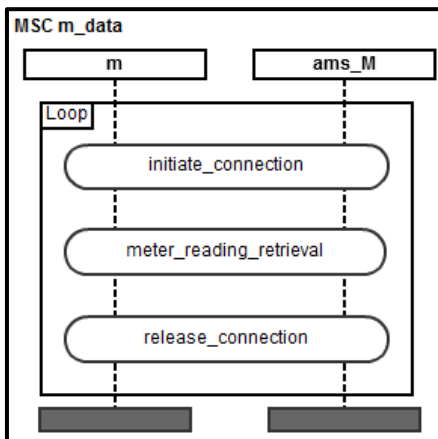


Figure 7: MSC showing the interaction between the users' smart meters and the AMS server

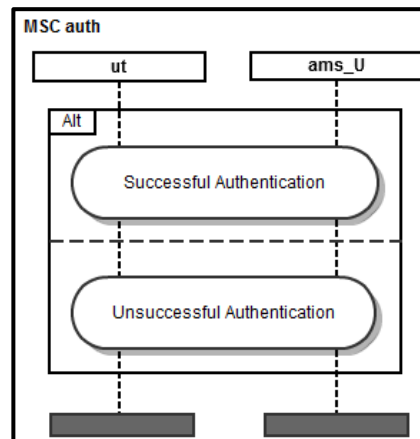


Figure 8: MSC showing the user terminal authentication

The collaboration between the smart meter and the AMS server encompassed connection initiation, data transfer and connection tear-down, as shown in Figure 7. The referred MSCs will be presented in section 3.3.

In Figure 8, the interface between a user terminal and the AMS server is studied, while the referred MSCs will be presented in section 3.4.3. Although the authentication scheme was still undecided, there was an apparent need for access control in this collaboration.

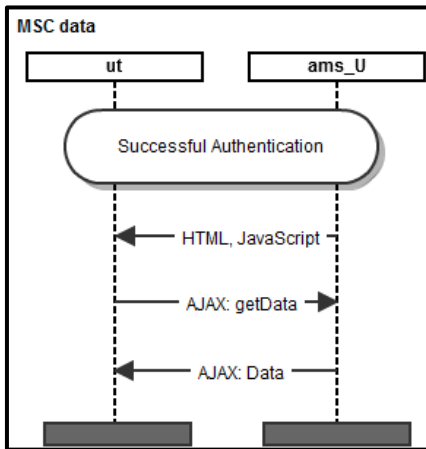


Figure 9: MSC showing data retrieval from the AMS server

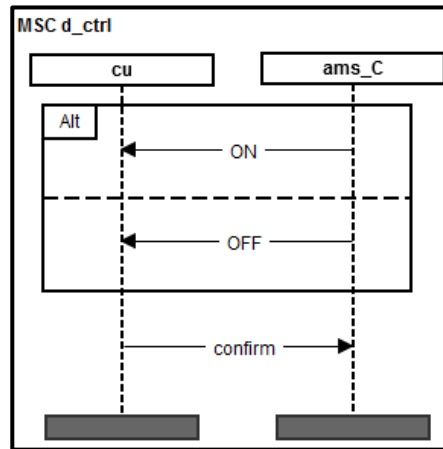


Figure 10: Device control is handled by the AMS server

In Figure 9 the MSC shows how the user terminal will retrieve data from the server and present it to the user. Figure 10 shows how the control unit is only dealing with commands originating from the AMS server. By doing so, one could deal with user requests on the server and perform the necessary authorization control at the core of the system, thus avoiding unauthorized access to controllable devices.

2.2.1.3 GUI Models

The importance of a well thought out graphical user interface cannot be overstated in applications meant to run on touch screen devices with limited screen resolution. The spatial restriction must be taken into account when deciding on the layout of buttons, menus and other graphical elements.

In this phase of the project, sketches of possible layout options were developed. The basis of these sketches was the requirements given by Wireless Trondheim. As with the system context model and the MSCs, the visual sketches helped were helpful in communicating ideas and alternatives concerning the GUI to other project participants.

The **first functional requirement** was as follows:

The application should have a start page. The start page should display information about the user's power consumption:

- a) Real-time consumption in KWh
- b) Today's consumption in KWh and NOK
- c) Display pricing information for next 24 hours
- d) Link to control page
- e) Link to historic data page
- f) Link to settings page

Based on these requirements, a GUI model of the start page was developed, see Figure 11.

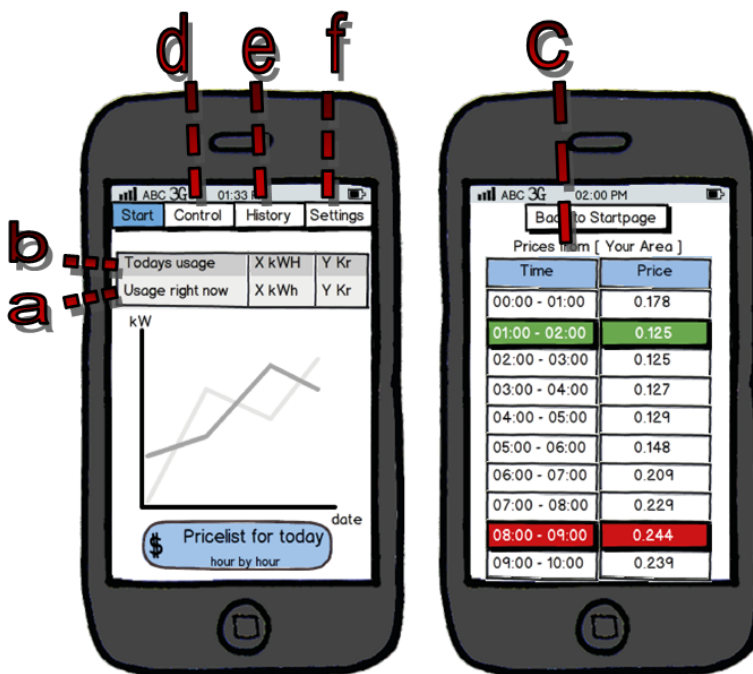


Figure 11: GUI model of the Start page and the pricelist for the next 24 hours

As can be seen from Figure 11, all of the requirements were integrated in the GUI model and these are marked with the corresponding letters.

The **second functional requirement** referred to the historic data page:

The application should have a separate page/tab, showing historical consumption data. This page should display to the user:

- a) *Historical consumption in KWh last year, month by month*
- b) *Historical consumption in KWh correlated with price, cost and temperature*
- c) *Consumption last month, day-by-day, in KWh and NOK*

As with the start page, a GUI model was developed for the historic data page to illustrate how the functional requirements would be realized. This model is shown in Figure 12.

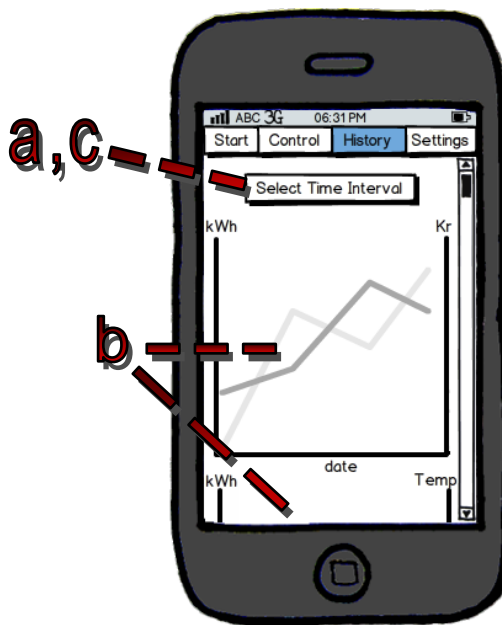


Figure 12: GUI model of the History page

It was decided that a graph view would be the best way of presenting the various correlation of data sources. Because of the requirements for the History page, there had to be made a space efficient solution to the problem of how data would be displayed for varying time intervals. The authors presented the option, shown

in Figure 12, where the user can manually set the start and end date for the graphs.

The **third functional requirement** referred to the device control page:

The application should have a separate page/tab where the client can control devices in the household:

- a) *It should be possible to switch on or off the devices in the list*
- b) *It should be possible for the client to set the indoor temperature*

After discussing alternative ways of realizing the aforementioned requirements, GUI models for the Control page were created, shown in Figure 13.

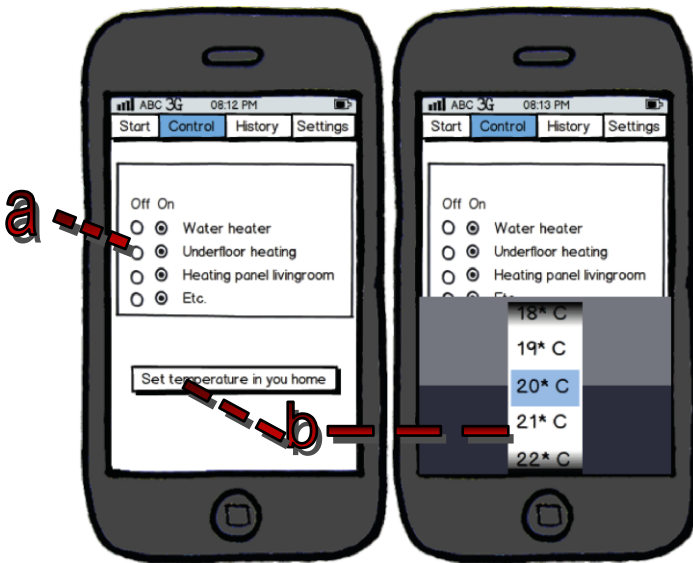


Figure 13: GUI models of the Control page

The **fourth functional requirement** referred to the settings page:

The application should have a separate page/tab for configuring personal settings:

- a) *Information about the current meter should be displayed*
- b) *The user should be able to choose between
 - i. *Application Languages*
 - ii. *Meters (if the user owns more than one meter)**
- c) *User should be able to end his current session*

To mirror these requirements, the model presented in Figure 14 was created.

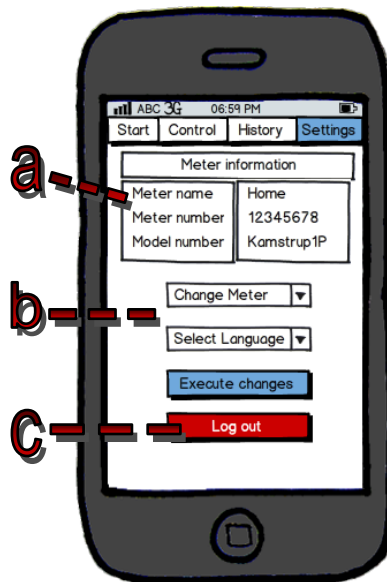


Figure 14: GUI model of the Settings page

In addition to the aforementioned functional requirements, **non-functional requirements** also needed to be taken into account:

- Secure transmission of data between the user terminal and the AMS server
- Unauthorized access to the individual user's controllable devices must be prevented
- The GUI should be intuitive
- The application should be responsive
- The design of the system should facilitate future development
- The application must be cross-browser compatible

Designing Functionality

As the functional and non-functional requirements were settled, the focus was directed towards developing functionality models mirroring the specified functionality. In Figure 15, the inner logic of the AMS Server block is shown. The different processes and the interactions between them are illustrated with boxes and arrows. Next to the arrows, the interface signal(s), surrounded by brackets, are declared. As seen from the block diagram, the AMS server consists of several separate processes that interact with the environment and the database. The *price_proc* and the *temp_proc* are both initialized by the *cron* daemon⁵ which is a time based job scheduler in UNIX systems allowing the execution of scheduled commands [12]. These processes retrieve data from the environment which are then stored in the database. This data is further used by the *application_proc* which interacts with the user terminals where the user will have the data graphically presented.

⁵ For an explanation on how the *cron daemon* is utilized in this system, see section 3.3.2

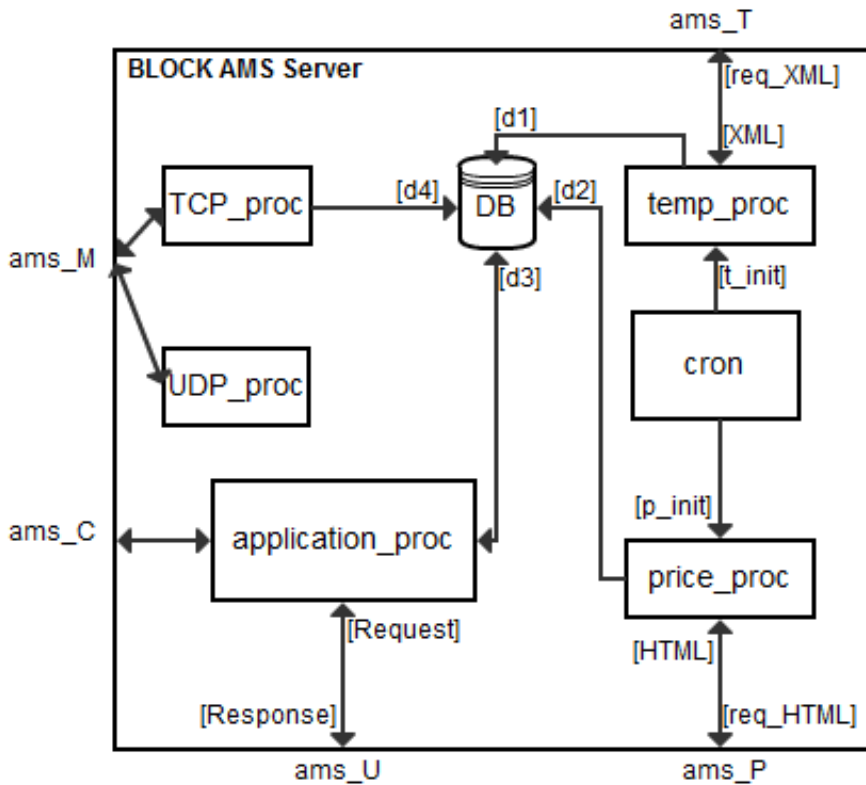


Figure 15: Block diagram of the AMS Server showing the inner processes

The main server logic is handled by the *application_proc*. It will receive user requests and based on the individual request, it will return either an HTTP or an AJAX-response. It also interacts with the control unit in the user’s home for enabling control functionality.

TCP_proc and *UDP_proc* both interact with the users’ smart meters. The meter data retrieval will be further discussed in section 3.3.1. The database is marked *DB* in Figure 15 and it receives data from all processes, except *UDP_proc*. The only process retrieving data from database is *application_proc*.

Next, the various processes were depicted in process diagrams, as shown in Figure 16 and Figure 17. The incoming and outgoing signals in these process diagrams can be recognized from the signals declared in Figure 15.

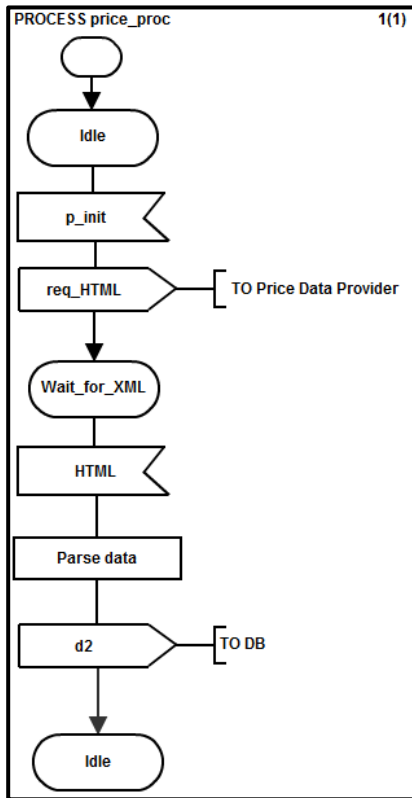


Figure 16: The behavior of the price process

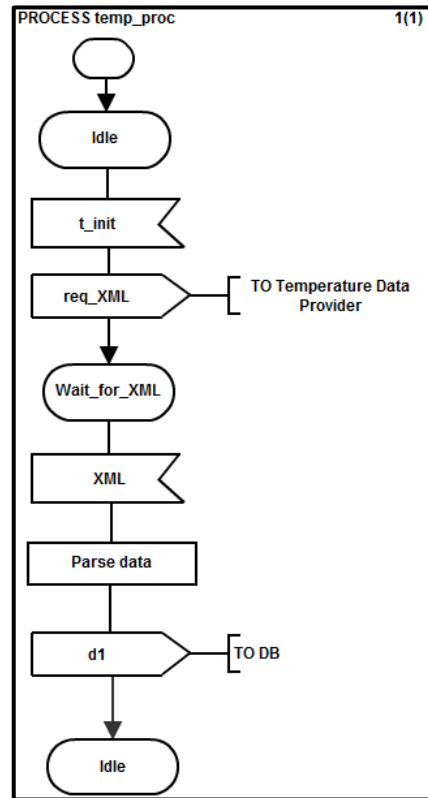


Figure 17: The behavior of the temperature process

These models served as a means to help plan and discuss inner-process behavior. The process diagrams complement the block diagram in Figure 15 nicely by addressing process-specifics, hence allowing a higher abstraction-level in the block diagram.

At this stage, the overall documentation now included:

1. **System context models**, showing the system entities and their relations
2. **User interface models**, illustrating GUI decisions that had been made
3. **MSCs**, depicting interfacial behavior
4. **Functionality models**, describing the behavior of the different system components.

With these models as the foundation, the system was ready to be implemented.

3. The Smarties system

For a user to feel that the gain of an AMS system is worth the cost, it is important to make him enthusiastic about the service. The application is where interesting data and facts can be presented to the user, and therefore it is important to make the application as user-friendly as possible. It must make the user feel that the changes he is making to his consumption patterns are reflected in the results presented to him. To achieve such an effect, the right technologies and ideas need to be incorporated in the system. This chapter will present the work that's been done to create an AMS web application prototype.

3.1. Similar Systems

Although the field of smart energy solutions is relatively new, it is not an unexplored field of technology. In this section some projects and products that share similarities with the system depicted in the Smarties project will be presented.

3.1.1. Google PowerMeter

The Google.org Project is the philanthropic part of Google which concentrates on using Google's vast base of knowledge and resources to help solve some of the world's global challenges [13]. One of these challenges is wasted energy. In order to help people get a better understanding and awareness about their energy usage, Google.org developed the iGoogle Gadget called Google *PowerMeter*. This free web application initially needed to work together with a user's smart meter in order to deliver energy usage information. Google simply provided the means of storing and presenting this information [14]. However, after the release of Google PowerMeter, several other companies have launched compatible

systems allowing the Google PowerMeter to function without a smart meter, two of which will be discussed further in section 3.1.2 and 3.1.3, namely AlertMe and TED.

As shown in Figure 18, from Google’s own PowerMeter page [15], Google PowerMeter provides the user with several tools for monitoring the energy consumption. Among these are graphs showing both the real-time usage, and the possibility of going back in time to get a graphical view of the past usage. Beneath the graph there is information about the total usage and an estimate of how much this usage will cost if the pattern is continued. This information is available on a day-by-day basis. In addition Google have created a way for the user to constantly monitor how his usage is today compared to the past.

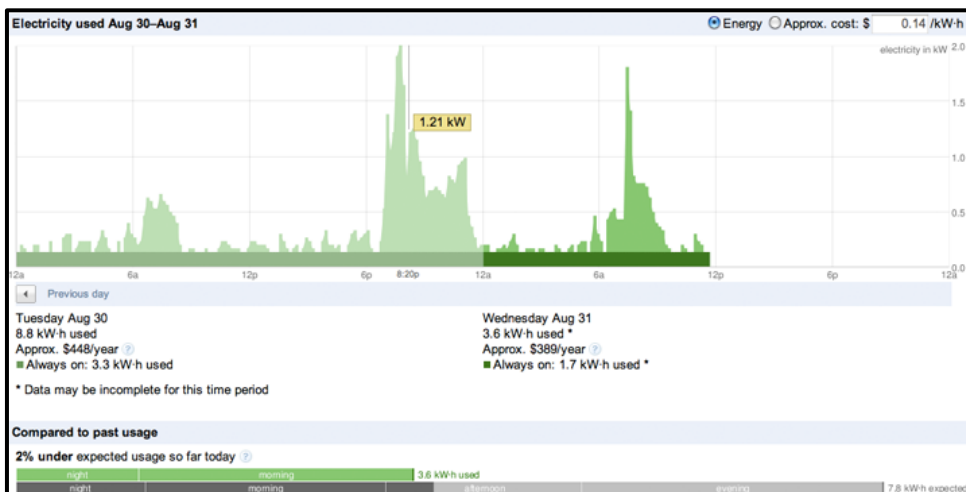


Figure 18: Google PowerMeter allowed people to monitor their power consumption [15]

Since the service was shut down on September 16, 2011 [15], this system will not be a competitor for the Smarties system, and it would not be possible to use these free services from the PowerMeter to build the Smarties system either. Even though this initiative has ended, it has helped promote this area of technology and motivated other to do further work in this field.

3.1.2. AlertMe

AlertMe is a British company that according to their website [16] have a vision to “empower consumers and give them unprecedented visibility and control of their homes wherever they are”. They offer a cloud-based smart home system providing an abundance of services ranging from security in the form of motion sensors and cameras, to SmartEnergy and appliance control functionality which are more important in relation to the Smarties project. The AlertMe architecture is illustrated in Figure 19 and shows how the system is divided into a home part and an internet part.

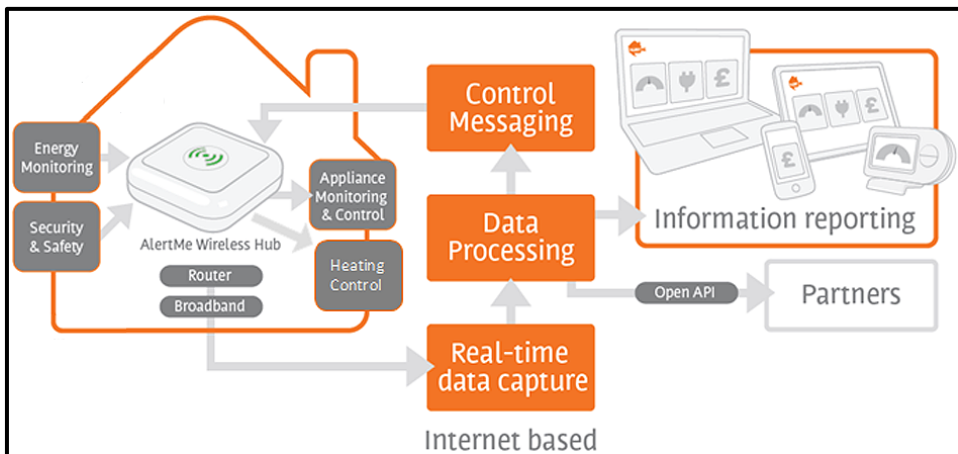


Figure 19: An overview of AlertMe’s system design [17]

The Wireless Hub creates a secure Home Area Network consisting of any number of sensors, controllable units and monitoring devices. These all connect to the Wireless Hub using ZigBee⁶, and from there the traffic goes via the user’s router to AlertMe’s servers in the cloud where all data is stored and processed in order to provide applications like the one shown in Figure 20. This figure is a snippet of the web application demo where the user allegedly can view a number of informative facts about his consumption in real-time for individual appliances as well as the total consumption. In addition, inside- and outside temperature and the possibility of controlling appliances in the house is shown (these last two

⁶ The ZigBee technology will be discussed in greater detail in section 3.2.3

functionalities are not shown in the snippet). This architectural design in Figure 20 is based on the same ideas as the Smarties project, but according to their own support service, this system will not function on a 3-phase system, or outside the UK. Therefore, the AlertMe system will not be a competitor to the Smarties project in Norway at this time. A complete answer from the support service is available in Appendix C – *Email correspondence with AlertMe*.

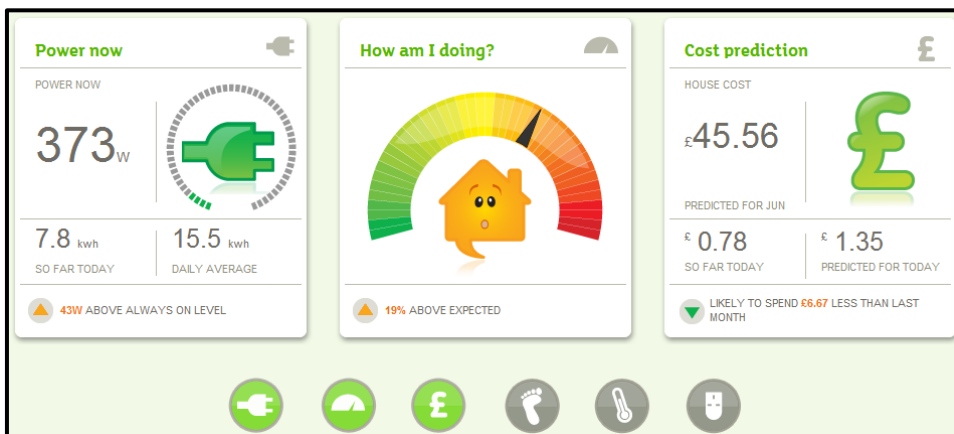


Figure 20: Detailed consumption information is presented in the AlertMe application [18]

An advertisement on their website [17] claims that “*AlertMe works with or without a Smart Meter, so we are uniquely positioned to support not only Utility and Smart Grid partners, but also Telecommunications operators...*”. According to their aforementioned support service this simply means that their own SmartMeter clamp will function even though the user has a normal smart meter installed in the home and will not interfere with it, not that the existing meter can take care of the collection of energy consumption data. Even though there are many similarities between AlertMe’s SmartEnergy system and the planned functionalities for the Smarties system, there are also some significant differences. One of the most important goals of the Smarties group is to propose a solution that will, according to the proposal template shown in Appendix A – The Smarties proposal template, “*potentially reduce the cost of AMS with 30-50 % utilizing existing infrastructure*”. AlertMe does not utilize the existing infrastructure, thus requiring the user to buy and install all the various AlertMe

equipment even though the user might already have a smart meter installed in his fuse box.



Figure 21: There is no need for a smart meter when using AlertMe [19]

AlertMe’s equivalent to the smart meter is shown in Figure 21 which is the *SmartMeter reader* that measures the current flowing through the cable it is clipped around. By doing so, AlertMe does not need to rely on smart meters from third party suppliers. However, it adds cost and installation efforts for the users who already have a smart meter, which is the customer base the Smarties group is aiming for. Since this way of measuring the consumed electricity is done by installing the measuring clamp around the cable instead of having the current pass through a measuring unit, the accuracy will never be as accurate as a regular smart meter. They claim that the accuracy of their SmartEnergy system “*should be at least 95%*”. This inaccuracy excludes this type of system to be used in conjunction with power companies to produce power bills in the same way the smart meters are to be used in the Smarties project.

Another difference between AlertMe and the Smarties system is the communication protocol used between the control unit and the control modules. AlertMe is using ZigBee, while it is expected that the Smarties system will be

using KNX. Both these protocols, as well as others, will be discussed further in section 3.2.3.

3.1.3. TED – The Energy Detective

TED [20] is another example of a smart energy system where the user does not need to have a smart meter installed in order to use the system. The TED system architecture is depicted in Figure 22 and it shows that the principle behind TED's measuring transmitting unit (MTU) is the same as AlertMe's smart meter clamp. It looks much like the one presented in Figure 21 as well, although it is not as easy to install. In order to install the MTU one needs to connect two cables inside the fuse box for the meter to be able to communicate with the gateway through power line communication (PLC), in addition to the clamp around the current cable that registers the power usage. The manufacturer claims that TED's MTU is accurate within 2 %, which is fairly good, but it will never be as accurate as a system using a smart meter. The gateway uses ZigBee to communicate with the handheld display shown in Figure 23, while Ethernet is used to connect with the user's computer or internet gateway. TED has also developed GUI for both computer and touch devices. However, unlike AlertMe, TED does not let the user track how much each appliance has used, only the total usage [21]. Also, TED systems offer no way of controlling appliances in the home, it is simply a monitoring tool.

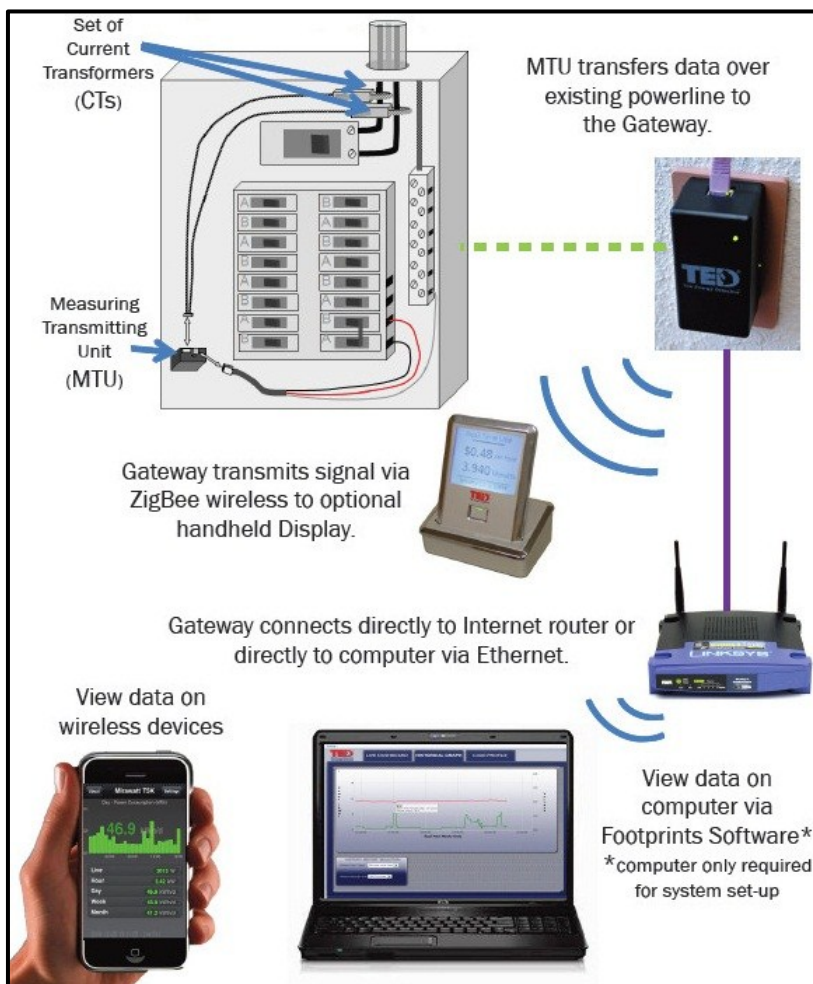


Figure 22: The TED system uses several different ways of communicating in order to transmit usage data from the MTU to the user’s devices [21]

It is clear that the installation is neither as easy, nor as safe as what one would want of an install-yourself-kit. It demands quite a bit more than a normal plug-and-play system, and at the same time requires the user to do work in his own fuse box. This fact is recognized by the creators of TED in their statement “*serious injury / death could occur if you are not familiar with electrical components and operation of the circuit breaker panel*” [19]. It is not considered good practice to tamper with the fuse box unless you are a certified electrician, additionally it might not even be legal many places. Furthermore, according to the installation guide [22], it is said that “...*if you have a 3-phase or 230V 50Hz*

electrical service, TED 5000 will not work”, which means that this system won’t even work in any of the Scandinavian countries. This type of problem is a good argument for why it might be better to use a system where it is required that a standardized smart meter is already in place.



Figure 23: TED’s wireless display presents the user with consumption information [20]

Unlike the other systems described in this section, including the Smarties project, TED does not store the usage data on servers in order to offer an online service where the data can be viewed. Data is instead displayed continuously on the wireless display, as well as the possibility of using a computer or a mobile device both locally and via the internet if the *TED gateway* is connected to an internet router. The data is stored locally in the gateway, with the possibility of downloading detailed data to a computer at any time using their own *Footprints software* [23].

3.1.4. Observation

One evident observation to be made from this research is the fact that none of these systems have the same setup as the Smarties system. Except from the discontinued Google PowerMeter, none of these solutions utilizes pre-existing smart meters, but requires the users to invest in additional equipment. It’s also

worth noting that these measurements are not as accurate as the ones registered by a regular smart meter. Smarties will offer a more accurate service at a lower price by utilizing the pre-existing smart meters and Wi-Fi in the users' homes.

3.2. Deciding on Technologies

To achieve the best possible solution, different design options were evaluated in order to select the most promising one for this particular system. This section elaborates on the different technology decisions that have been made. Figure 24 shows the various system entities that are directly involved in delivering the application to the users.

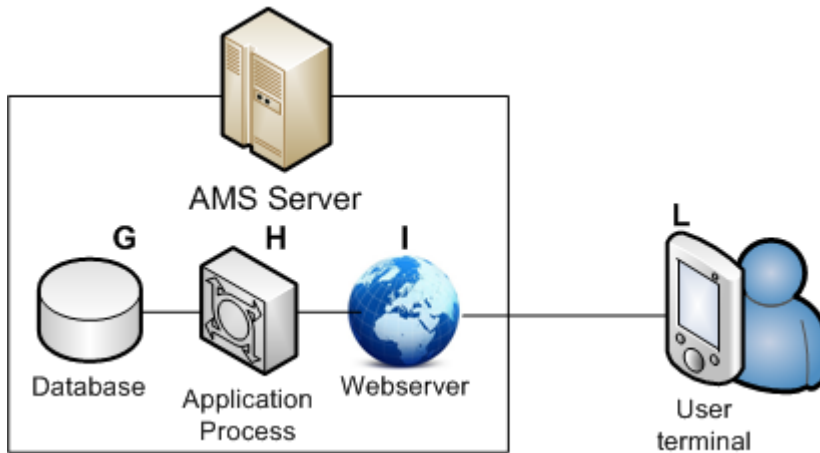


Figure 24: The web application part of the Smarties system

3.2.1. Server-side

There are numerous alternatives to choose from when deciding on a server-side programming/scripting language. Java, ASP.NET, PHP, Ruby and Python, which are some of the most common server-side programming languages, all offer useful features.

The decision to choose a certain programming/scripting language is often determined by personal preferences. The different languages offer, to a great

extent, equivalent features, but what is distinguishing one language from another is often the amount of code or time needed to implement a task, in addition to the readability of the resulting code.

WT's server runs Ubuntu, which meant that Microsoft's ASP.NET had to be excluded. Due to limited experience with both PHP and Ruby, these options were also discarded, leaving Java and Python as remaining alternatives.

3.2.1.1 Java

The Java platform is composed of the Java Virtual Machine and the various APIs shown in Figure 25. By letting the Java code run on a virtual machine and not directly on the OS, it is interpreted into code readable to the OS. This feature makes Java code portable across different operating systems. However, the virtual machine requires Java source code to be compiled into bytecode prior to execution.

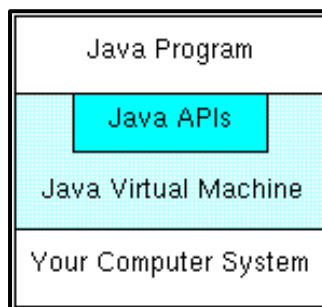


Figure 25: The Java platform [46]

Java Servlet is the API enabling Java to be used for creating web services. As a means of creating interactive web applications, Java Servlet has gained popularity. According to Hunter and Crawford [24], servlets differ from regular Java programs in how they don't have *main* methods. As an alternative, servlets are services invoked by the server once a request is being processed. Figure 26 shows how servlets handle incoming HTTP requests.

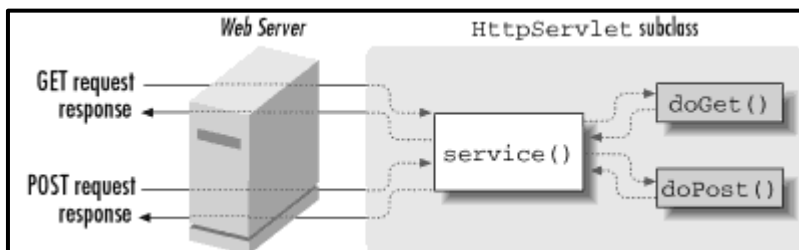


Figure 26: A Java servlet handling HTTP requests [24]

3.2.1.2 Python

According to the official Python web site [25], Python is a “*remarkably powerful programming language that is used in a wide variety of application domains*”. The Python website claims that compared to other programming languages such as Java, Python has some differentiating traits. Among the features mentioned are clear and easily read syntax, exception-based error handling, high-level dynamic data types and intuitive object orientation.

As opposed to Java which is a compiled language, Python is an interpreted language [26]. Code written in an interpreted language is translated into bytecode during runtime and not compiled prior to being run. In practice, this means that Python source code can run directly, without the need of an executable file. Python and other interpreted languages therefore facilitate shorter development and debugging cycles, compared to compiled languages. However, due to the load times of interpreted languages being higher than those of the compiled ones, Python combines the two techniques. The first time a Python program is interpreted, the interpreter compiles the program into bytecode [27]. The next time this program is executed, the compiled version is loaded, hence reducing the application’s loading time.

To streamline the creation of Python servers, a vast number of web frameworks have been developed for this language. A web framework offers a pool of APIs and modules which facilitates developers in writing web applications by abstracting lower level details such as transport protocols, sockets or thread control. By outsourcing the lower level tasks to the framework, the developers can focus on getting the application logic right. As a result, the time spent on prototype development is reduced.

Based on the information presented in this section, it was decided to use Python as the server-side programming language. Since the features offered by Python and Java are equally satisfying, the fact that Python facilitates shorter development and debugging cycles resulted in Python being chosen due to the short submission deadline for the finished prototype.

3.2.1.3 Deciding on a Framework

There are numerous Python web application frameworks to choose from [28], where the following two are among the most popular:

- Django
- Pyramid

Django

Django is a high-level framework which supports fast development and clean design [29]. It is an established framework launched in 2003. According to Kaplan-Moss [30], the motivation for creating Django was the demand for an application framework supporting new features and new applications to be developed quickly.

Pyramid

According to Chris McDonough [31], Pyramid’s primary goal is to “*make it easier for a Python developer to create web applications*”. To do that, it provides the following traits:

- **Minimalistic** – Pyramid focuses on the core challenges in web development and leaves out excess features
- **Simple** – No need to know every detail of the framework in order to produce results. Suitable for creating working prototypes quickly
- **Well documented** – Every aspect of Pyramid is well documented
- **Fast** – Provides short execution times for common tasks
- **Reliable** – Every release is thoroughly tested
- **Open source**

Performance tests have shown that Pyramid performs slightly better than Django [32]. The leftmost graph in Figure 27 shows how the two frameworks performed the task of returning a “Hello World” response upon receiving an HTTP request. The graph in the center represents the performance for the task of returning an

HTML template, whereas in the rightmost graph the task was retrieving data from a database and returning the data as an HTTP response.

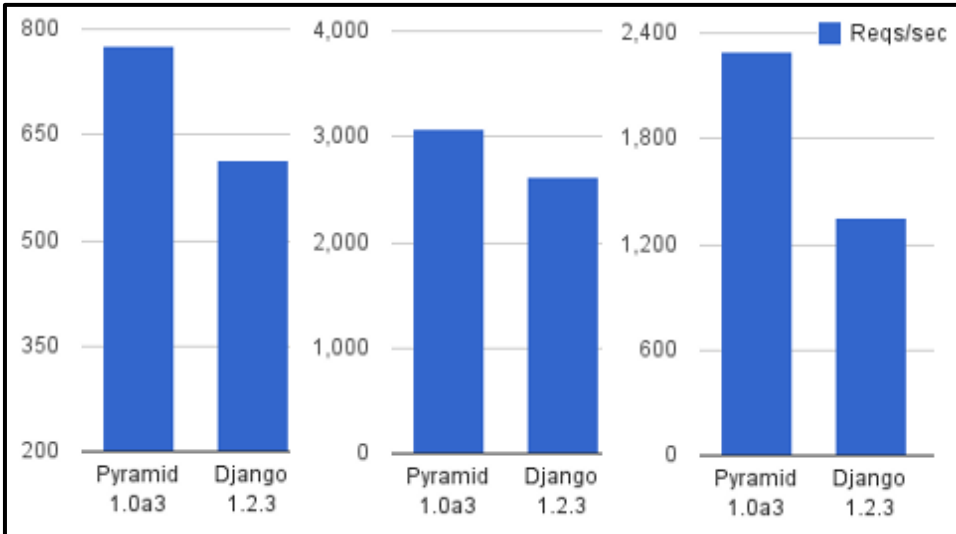


Figure 27: Separate tests showed that Pyramid performed better than Django for common web application tasks [32].

Based on the features and the results from the performance test, it was decided to implement the server-logic using the Pyramid framework as a foundation. Since WT uses MySQL databases for their current services, it was decided to also use MySQL for the Smarties system.

3.2.2. Client-side

There are several possible technologies to choose from when planning a web application. The decision was based on the application requirements, as well as the developers' personal preferences. One of the non-functional requirements was that the application needed to be compatible with the most common browsers and smart phones. In addition, the chosen technologies would have to support swift development due to the short deadline for finalizing the working prototype.

3.2.2.1 Using Purely Server-Side Logic

One possible solution, that was early discarded and will therefore not be discussed in great detail, was to implement the logic on the server side only. In that way the user terminal would not do any processing and hence the performance requirements for user terminals would be considerably reduced compared to a solution where scripting was done on the device itself. In that way the performance properties of a user device would not prevent the use of the application. However, most devices today, both handheld and desktop, have sufficient processing power to run an application of this magnitude. By using purely server-side logic, it would mean that for every little update needed on the page, the server would have to create a new HTML document and send it to the device, hence losing speed and dynamical features as opposed to an application running logic also on the client.

The application would be more responsive and user-friendly with logic on the client-side that enable single fields in the HTML document to be updated without the need for a complete reload of the page. Since this is meant to be an application for mobile devices, it needs to be taken into consideration that users might use 3G/4G, thus the importance of keeping the network traffic to a minimum. Purely server-side logic would result in more network traffic compared to a solution where logic is implemented on the client as well. These reasons combined resulted in this approach being discarded.

3.2.2.2 Why JavaScript?

When deciding on a client-side scripting language, it's an unavoidable constraint that browsers only support JavaScript as the scripting language. There has been developed several other scripting languages that can be used, but they all must be compiled into JavaScript code that will run in the browser. Examples of these types of languages are CoffeScript, ClojureScript, Dart and haXe to mention some [33].

A completely different option would have been to use the *Adobe Flash* [34] multimedia platform to create the web page. Flash is a widely used platform for making animations on the web, and can be displayed in every browser that has

downloaded and installed the free Adobe Flash Player. In addition, Adobe has created a JavaScript equivalent scripting language called *ActionScript* [35] which is also used to create applications and websites, but only for the Adobe Flash Player platform. Adobe Flash could have been used as the client-side scripting language for the Smarties application, had it not been for the fact that Adobe Flash is not supported on any of Apple's mobile devices. It was simply not an option to make an application incompatible with iPhone and iPad, leading to Flash being discarded.

Due to the fact that Flash would not work on iOS devices and that the aforementioned JavaScript alternatives would have to be compiled into JavaScript, both lead to JavaScript being chosen as the client-side scripting language. The developers' experience with JavaScript also contributed to this decision.

With JavaScript selected as the client scripting language, JavaScript Object Notation (JSON) was chosen as the data-interchange format for transferring data from the server to the user terminals. JSON can easily be used in conjunction with AJAX for requesting text-based data from JavaScript. In addition, JSON is easy to read and write for humans as well as easily generated and parsed by machines, making it an ideal match for the AMS web application.

3.2.2.3 Why jQuery Mobile?

Since JavaScript was chosen as the appropriate client-side scripting language for implementing the client-side functionality for the AMS web application, there was a need for a suitable JS framework. A JS framework contains sets of useful functions and is very useful in order to ease JS development of interactive websites. As mentioned in section 1.4 about web applications, when Web 2.0 were introduced with a more user-centered approach, the way web pages were designed naturally changed as well. As a result, several new technologies emerged.

One such JS framework created to help developers construct web applications tailored for mobile touch devices is Sencha Touch 2. According to their website [36], Sencha Touch is built to make fast and well-functioning mobile web

applications using their own JavaScript framework called Ext JS 4 [37] as the basis of the development process. They brag about their product, saying it's "*the only framework that enables developer to build fast and impressive apps that work on iOS, Android, BlackBerry, Kindle Fire, and more*" [36]. While that statement might well be close to the truth, the fact that their system does not work in Internet Explorer (IE) makes it inappropriate for the Smarties application. Even though statistics [38] show that the use of IE is decreasing, it is still at 18.3% in April 2012, and that is far too big a market share to just ignore completely.

Another JavaScript framework designed for mobile devices is jQuery Mobile, which is a relatively new concept developed by the jQuery Team. It was first announced on August 13th, 2010 [39] and the first stable version was released on November 16th, 2011 [40]. The motivation for developing this mobile version of jQuery was the growing desire for an easy way of designing and developing web-applications tailored for mobile devices. The jQuery Project describes jQuery Mobile as "a unified user interface system that works seamlessly across all popular mobile device platforms, built on the rock-solid jQuery and jQuery UI foundation" [41], where jQuery is the actual JavaScript framework. In other words, jQuery Mobile is a large set of functions, features and animations meant to aid the JS development process in building mobile web applications. This makes jQuery Mobile perfect for web applications meant to run on touch devices since it adds graphics and functions tailored for such devices, but at the same time it can be used in browsers on desktop computers as well.

According to the jQuery Foundation [40], the jQuery Mobile framework provides means of developing HTML5 based web-applications by providing CSS offering modern styling designs, giving the developer the possibility of using the newest within web technology in the making of web-applications. All the most common smartphones, tablets, e-readers and desktop browsers are supported in jQuery Mobile version 1.0; Apple iOS 3.2-5.0, Android 2.1-2.3, Android Honeycomb, Windows Phone 7-7.5, Chrome Desktop 11-15, Firefox Desktop 4-8 and IE 7-9 to name a few.

jQuery Mobile was chosen since it has all the necessary functionalities to enable the creation of applications that would both work and look good on touch devices as well as desktop computers browsers. In addition, it is a new and exciting technology with many useful features, and therefore a very interesting technology to explore.

3.2.2.4 Native Application

It was decided from the start that the Smarties application should be a web application, but it is interesting to show how easily the system could be reused if a native application is to be developed in the future. An application is considered to be *native* when the application is made to run directly on the device OS and not run within the terminal's browser. The difference between Figure 28 and Figure 24 is that the webserver is no longer needed if the application is a native application on the user terminal. In a system like the one shown in Figure 28, the native application will request only necessary data from the server process. Those data would then be used to draw graphs and present the information on the terminal device in the same way as the web application would do, only by using the device-specific technologies instead of JavaScript, HTML etc.

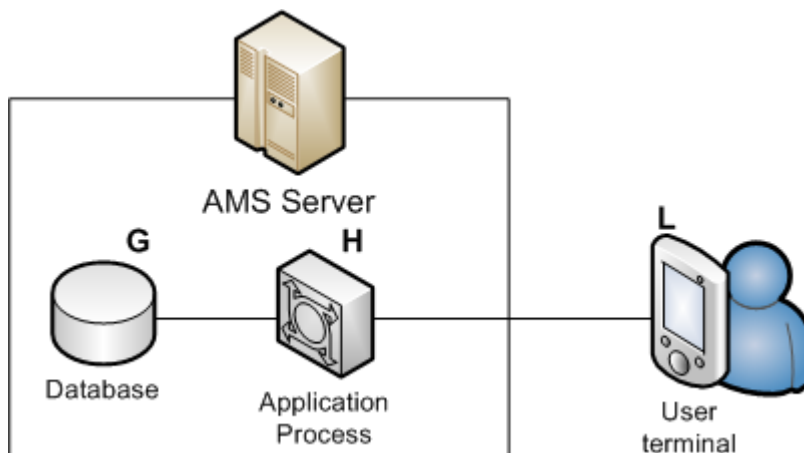


Figure 28: The Smarties system could easily be changed to support native applications

Since it was decided not to develop the prototype as a native application it meant that even though the user terminal is a touch device, the application must be opened and run in the browser. This is not as practical as a native application,

and therefore the code presented in Figure 29 was added to the HTML file in order to make the web application appear more like a native application.

```
<meta name="apple-mobile-web-app-capable" content="yes"> 1  
2 <link rel="apple-touch-icon-precomposed"  
    href="styles/images/shortcutIcon2.png"/>
```

Figure 29: Code snippet to make home screen bookmark on iOS devices

Unfortunately this code only works on Apple’s iOS devices at this time, and the code is triggered when the “Add to Home Screen” button in the browser is pressed. The two marked HTML elements in Figure 29 have the following purpose:

1. Hides the Safari user interface components (address bar at the top and navigation buttons at the bottom), making the web application look much like a native application, as seen in Figure 38. It still uses the browser to run the application, but it is run in an independent instance of the browser as a standalone application
2. Adds a pre-defined application icon to the home screen of the device

3.2.3. In the Home

There exist many different standards and specifications on how to realize communication between devices in a smart home. Selecting the most appropriate one is not always an easy task, and therefore some of the most used and recognized methods will be presented and compared in this section.

3.2.3.1 KNX

According to the KNX Association’s website [42], KNX is the Worldwide Standard for Home and Building Control. In fact, KNX is very much recognized as the worldwide standard as it has been approved by a number of international organizations of standardization in Europe, China and USA, as well as ISO. The KNX standard has evolved from predecessors such as EIB, EHS and BatiBUS [42] and through projects like these, the KNX Association has over twenty years of experience in the Home Automation market.

KNX provides a common language for Home and Building Control. By standardizing the communication between components compatibility amongst equipment produced by diverse manufacturers is ensured. In addition, it supports a broad range of communication mediums such as twisted pair, Wi-Fi and power line which makes it applicable in most homes.

In a KNX network the bus devices are either sensors or actuators. The range of use is listed below:

- Lighting
- Blinds and Shutters
- Security Systems
- Energy Management
- Metering
- Etc.

KNX offer ways of both controlling and monitoring these different functions without the need for a separate control center, which may be the case in other Home and Building Control Systems.

3.2.3.2 ZigBee

ZigBee [43] is a specification of high-level protocols designed to be a low-power, low-cost wireless communication method between different devices in one single control network. According to the ZigBee Alliance's specification overview [44], it utilizes the 2.4 GHz radio frequency band and the IEEE 802.15.4 standard in order to deliver a wireless personal area network (WPAN) where devices can communicate and interoperate independent of the supplier. Security and network layers are added on top of the IEEE 802.15.4 standard, as well as an application framework. However, there is not necessary interoperability between devices from different manufacturers since they have the possibility to create their own specific profiles on top of the ZigBee specification. This results in less interoperability since several manufacturers choose this option to integrate ZigBee in their own proprietary systems.

3.2.3.3 Z-Wave

Z-wave is quite similar to ZigBee in the way that it can also be used to create wireless smart home systems using the same type of low-power radio waves as the transport medium in order to remotely control appliances in the home. This type of communication medium is perfect for a home system since it can easily traverse household obstacles like walls and floors [45]. However, a difference is the radio frequency the two protocols use. With ZigBee using the same frequency range as Wi-Fi, namely the more crowded 2.4GHz band, Z-wave have chosen to use frequencies around the 900MHz band instead [46].

3.2.3.4 Comparison

The Dubai based company Shifra Smart Home Automation has carried out a study [47] of five different protocols for smart home communication, among them the three mentioned protocols. By defining the following requirements, a basis for comparison between them can be made:

1. Reliable communication
 - a. Does the system make sure that messages are received properly between the control unit and the control modules, and if not are they re-sent?
2. Secure communication
 - a. Are necessary precautions taken in order to ensure secure communication between the devices?
3. Features and capabilities
 - a. Is it possible to integrate this system in other in-home systems, like multimedia entertainment systems?
4. Investment protection and standardization
 - a. Is the protocol built on standardized technology ensuring that system entities can be repaired/changed by a different company at a later time without having to replace the entire system?
5. Interoperability
 - a. Can the control unit communicate with several control modules from different manufacturers and still function as intended?

	KNX	ZigBee	Z-Wave	X-10	En-Ocean
Reliable	✓	✓	✓	No***	No
Secure	✓	No*	✓	No****	✓
Features	No	✓	✓	✓	No*****
Standardized	✓	✓	✓	✓	✓
Interoperability	✓	Sometimes**	✓	✓	✓

*Weak hashing algorithms have led to it being penetrable by using software tools like “Killerbee”.

**As mentioned in section 3.2.3.2, the manufacturers have the possibility to create their own profile on top of ZigBee that will ruin the interoperability.

***Uses mostly PLC which is prone to interference in a home. Messages are not acknowledged and therefore not resent if lost.

****Since PLC is used, everyone who can tap into the house’s power-line can read and send commands as they wish.

*****It is supported, but not on the marked at this time

According to this study by Shifra, it is clear that in relation to these given requirements the Z-Wave protocol looks like the most promising candidate. However, there are other aspects to consider in the process of choosing which technology to utilize in a new project. On the grounds of the signs received from Bravida, they are most likely choosing KNX as the home control protocol on the basis of it being a widely used, well documented protocol which Bravida has prior knowledge about. At the same time it has not been taken into account that the system created in the Smarties project should be able to connect to multimedia entertainment systems, or other similar preexisting smart home systems for that matter. If this is a smart strategy is hard to say, but having the possibility to easily integrate the Smarties system with a potential customer’s preexisting system could easily be a good selling point and should probably have been taken into account in the decision process.

3.3. Data Retrieval

As shown in the system context diagram in Figure 6, the Smarties system will retrieve data from three different sources:

- The users' smart meters
- A temperature data provider
- A price data provider

In this section, the different solutions will be discussed and reasons will be given for the various decisions that have been made.

3.3.1. Meter Data

As depicted in Figure 7, the collaboration between the smart meter and the AMS server would involve a connection setup, followed by a data transfer sequence, ended by a connection tear-down procedure.

At the time the high-level sketch was developed, the implementation details were still undetermined. However, it was soon evident that one needed to combat a challenge, namely the Network Address Translation (NAT) functionality found in most routers.

According to the RFC1631 [48], NAT is a router function that enables addresses inside local domains to be reused in other local domains. This was originally a way of conserving IPv4 addresses. RFC1631 also states that any router running NAT should never advertise its local networks to the backbone. Only networks with global addresses may be known outside the local domain. The borderline between the local and the global address domain is illustrated in Figure 30.

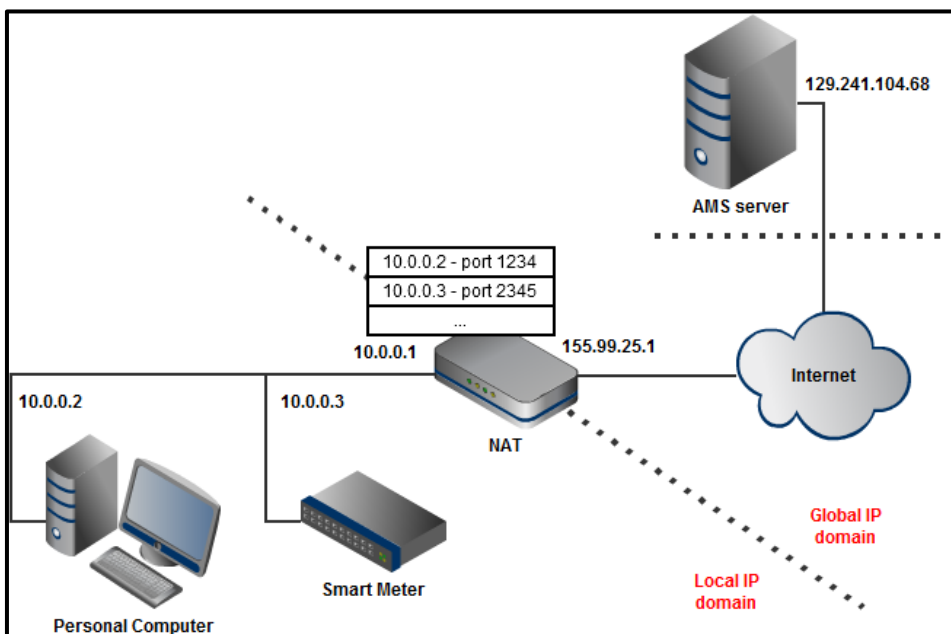


Figure 30: NATs represent the borderline between local and global address spaces

For the Smarties system, this meant that the AMS server had no way of knowing the IP address of the various smart meters, since these would connect to the users' private LANs, residing behind NATs. In fact, even if the AMS server did possess the address of the smart meter, the NAT would block any incoming requests. To solve this issue, the solution was that the smart meters would be the entity initiating the connection, thus “punching a hole” in the NAT.

UDP *hole-punching* is a technique for establishing two-way UDP connections between devices on opposite sides of one or more NATs⁷. According to RFC5128 [49], the UDP hole-punching works like this: After a NAT device (a router) has a record of having sent a packet to the destination, any packets originating from this IP address and port number will be let through. By initiating connection from the inside of the NAT, the server can use the received datagram's source address and port number for communicating with the smart meter.

⁷ It is worth noting that UDP hole-punching will not work with symmetric NATs [58]. These types of NATs are typically found in sizeable corporate networks.

The MSC references in Figure 7 could now be presented as stand-alone MSCs, shown in the following figures:

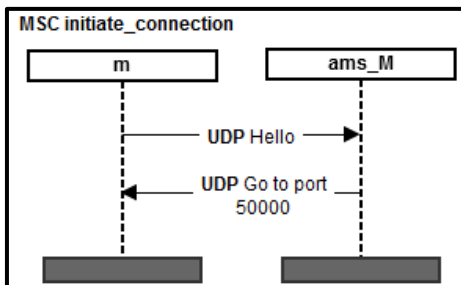


Figure 31: UDP hole-punching and first server-response

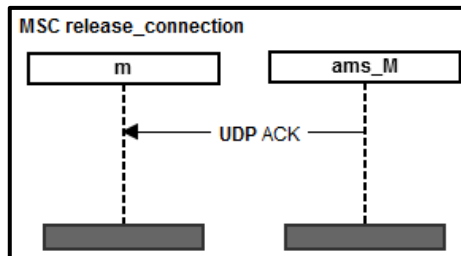


Figure 32: UDP ACK on initial HELLO

Figure 31 shows the initial hole-punching performed by the smart meter. The UDP response from the server instructs the smart meter to initiate a TCP connection on port 50,000. Figure 32 shows the final ACK returned to the smart meter after a successful meter reading retrieval shown in Figure 33. Upon receiving this ACK, the smart meter will return to its initial state.

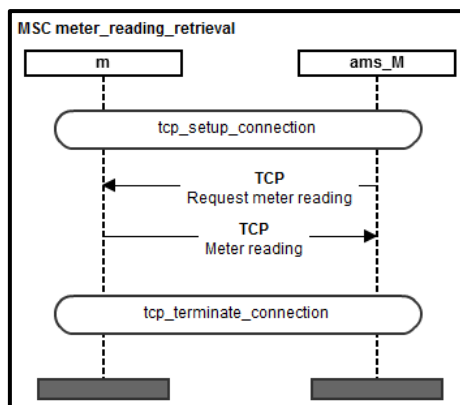


Figure 33: The meter reading sequence when consumption data is transferred to the server

Figure 34 and Figure 35 shows a TCP connection setup and a TCP connection tear-down, respectively.

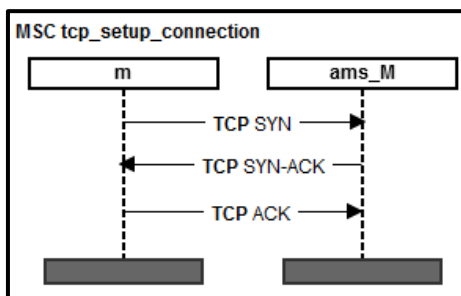


Figure 34: TCP three-way handshake

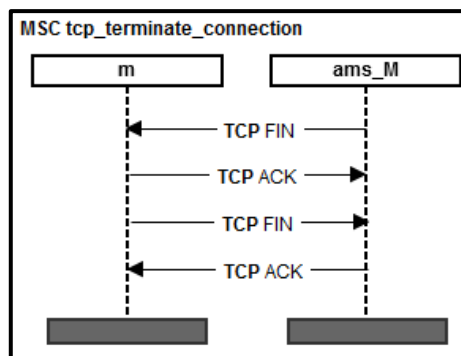


Figure 35: TCP connection termination sequence

To handle this communication, both a UDP process and a TCP process were created on the AMS server. From Figure 15 one can see how both these processes are independent from the AMS web application itself. These processes interact solely with the smart meters and the database.

The content of the *Meter reading* packets shown in Figure 33 are these two measurements registered at the time of the request:

- The total usage in KWh
- The current usage in KW

These two values are stored in the database along with the meter id of the smart meter as well as the timestamp. These data will be used later on to present the users with valuable information about their consumption.

3.3.2. Temperature and Price Data

By correlating outdoor temperatures with consumption data, the idea was to generate interesting user observations. It would give the user an impression of how much impact heating and cooling have on the electricity bill. Such impressions might motivate users into buying more energy efficient appliances or invest in improved insulation. Temperature data is available as XML files at the meteorological institute's web site. A temperature process, shown as *temp_proc* in Figure 15, was created on the server. Using an XML parser, this

process extracts the data of interest from the retrieved XML files and stores it in the database.

Price data is needed to give users economical motivation for assessing their consumption patterns. By combining hourly price information with hourly consumption, cost could be calculated for every hour of the day. Price data was retrieved from Nord Pool Spot's website as HTML documents. A temperature retrieval process on the AMS server was created, shown as the *price_proc* in the block diagram from Figure 15. Upon receiving the HTML documents, this process extracts the information of interest and stores it in the database.

Common for both the temperature and the price process is the way they are initiated. The operating system on the AMS web server is UNIX-based, offering the possibility of creating *crontabs*, which are scheduled commands executed by the *cron daemon* shown in the AMS server block diagram in Figure 15. The *crontabs* created for the price and the temperature process, are configured to execute at the same hour of every day, thus keeping the price and temperature data up to date.

3.4. The AMS Web Application

With GUI- and functionality models at hand, the implementation of the application could start. The models were mostly followed, but as the system development process in Figure 5 indicates, even after the implementation process had started, modifications were made to the designs when new ideas or unforeseen problems occurred. The specific files that will be used in the explanation of the system are *index.pt* and *dataFetchAndDraw.js*. The *index.pt* file is the server-side page template file that is used to generate the HTML document sent to the user terminal. The *dataFetchAndDraw.js* is the JavaScript file that is sent to the user terminal along with the HTML document. This JS file runs on the user terminal and its job is to fetch data from the server and dynamically present it in the application.

This chapter will discuss the functionality and appearance of the AMS web application from the data collection to the presentation on the user terminal. Section 3.4.1 explains the layout of the application's client-side as well as how it works. More technical details regarding the application's functionalities will be explained in sections 3.4.2 to 3.4.6. Testing and optimizing measures will be discussed in section 3.5.

3.4.1. Presenting the Application

Using the models and designs presented in section 2.2 as a basis, the practical implementation of the web application was accomplished using the technologies presented in section 3.2.

3.4.1.1 jQuery Mobile Page Scripting

A great feature in jQuery Mobile is the page scripting [50] which takes advantage of an AJAX navigation system which overrides the browser's default navigation system. Normally when a link is pressed the browser will be redirected and load the full page. jQuery Mobile reformulates this request to act as an AJAX request, which means that instead of redirecting and loading a completely new page, only the required parts of the HTML document is returned

and presented to the user. For this reason, the entire application consists of only one HTML file where jQuery only displays separate parts of the file at a time.

```
<div id="start" data-role="page">
  <div data-role="controlgroup" data-type="horizontal">
    <a id="index" href="#start" data-role="button">Start</a>
    <a id="styrEnheter" href="#control" data-role="button">Control</a>
    <a id="historie" href="#history" data-role="button">History</a>
    <a id="innstillinger" href="#settings" data-role="button">Settings</a>
  </div>
```

Figure 36: Each "page" is given a different id and this id is used for browsing between the different pages

Figure 36 shows a section of the HTML code for the Start page, and the jQuery specific part of this is present already in the first line with the *data-role="page"*. When a user clicks the Start page link while browsing on another page of the application, jQuery simply replaces the part of the document that is presented to the user at that time with the code inside the *<div>* that has *id="start"*. For this reason, the HTML file is only loaded once which in turn also means that the *head* is only executed once with the JavaScript and CSS files only being loaded once. This saves both processing and transmission time on the user terminal.

```
$('#start').live('pageshow', function() {
    // ... code
})
```

Figure 37: Each page element from the *index.pt* file has a corresponding function in the *dataFetchAndDraw.js*

The regular way of activating the JavaScript file in jQuery is to use the *\$(document).ready()* function which will run the content of the function as soon as the HTML is loaded and ready. In jQuery Mobile this does not work since the different "pages" are simply separate sections of the same document, hence the function would only run after the first time the HTML file was loaded and not for every page-change within the file. Instead every page element is given its own function in the *dataFetchAndDraw.js* file where the page-specific code will automatically run when that page element is loaded. Figure 37 shows how this

function is made for the Start page, meaning that when the Start page portion of the HTML file has been presented in the browser, the code inside this function will automatically run.

3.4.1.2 Login Page

The first HTML page presented in the browser when entering the application is the login page shown in the illustration to the left in Figure 38. If the username and password is authenticated, a session is opened and the user gains access to the application. If not, the error message shown to the right in Figure 38 is displayed and access is denied. The details behind the session functionality will be described in section 3.4.3.



Figure 38: The Login page

3.4.1.3 Start- and Pricelist Page

The first thing a user sees after logging in is the Start page, and therefore it was important to include a selection of the most important information here. In Figure 39 the entire Start page is shown, and it is clear that all the functional requirements presented in section 2.2.1.3 are fulfilled. Usage information in the form of both a table and graphs, as well as links to all the other main pages of the application have been implemented. The top part of the price information page is presented in Figure 40. In addition to giving a list of hour-by-hour prices for the next day, the highest and lowest prices are highlighted, as well as showing the average price of the day. By having this information one day in advance, a user has an opportunity to plan his power-consumption in order to save money.

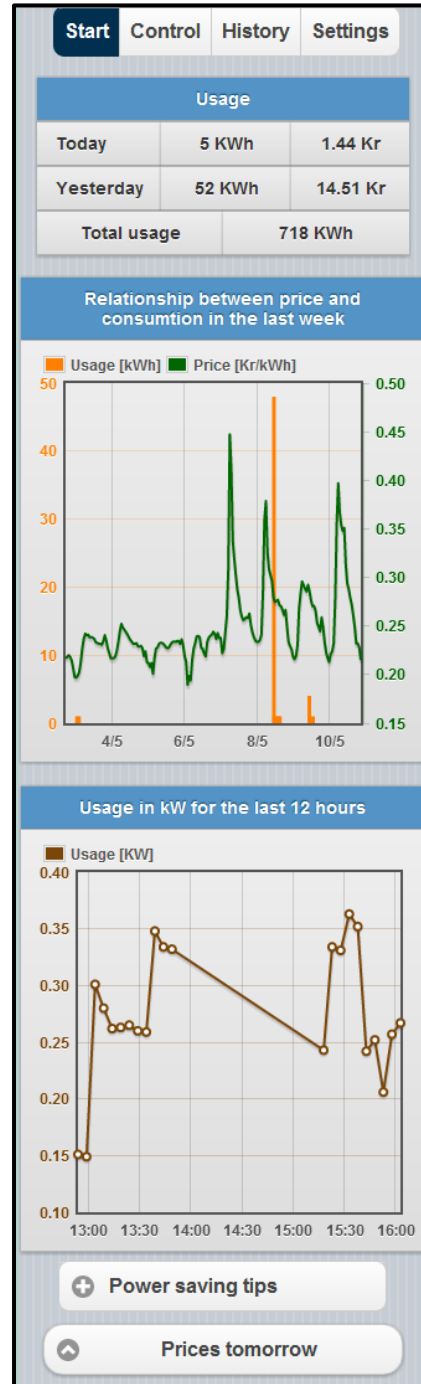


Figure 39: The Start page

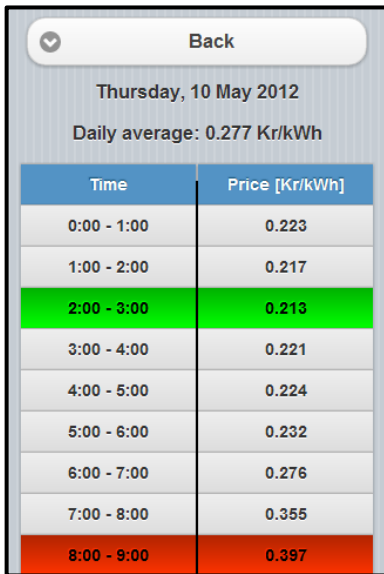


Figure 40: Prices for the next day is available in a separate page **Figure 41: Power saving tips are presented on the start page**

As stated earlier, the motivation for developing this application is to get the users to save electricity as a consequence of being presented with their consumption. To provide tips on how to be more energy efficient, the collapsible list *Power saving tips* was added to the Start page shown in Figure 41. The last four entries in the list contain general tips on how the users can make their homes more energy efficient. The *Personal usage patterns*, shown in Figure 42, generate user tips based on the user’s consumption data. By differentiating consumption during night-, work- and after-work hours, unnecessary power usage can be identified. The data shown in Figure 42 are real consumption data collected from a pilot user, but as mentioned in section 1.1, the smart meters from Kamstrup are a new trial product specially made for the Smarties project, so there are still some irregularities in the data that is collected at this time.

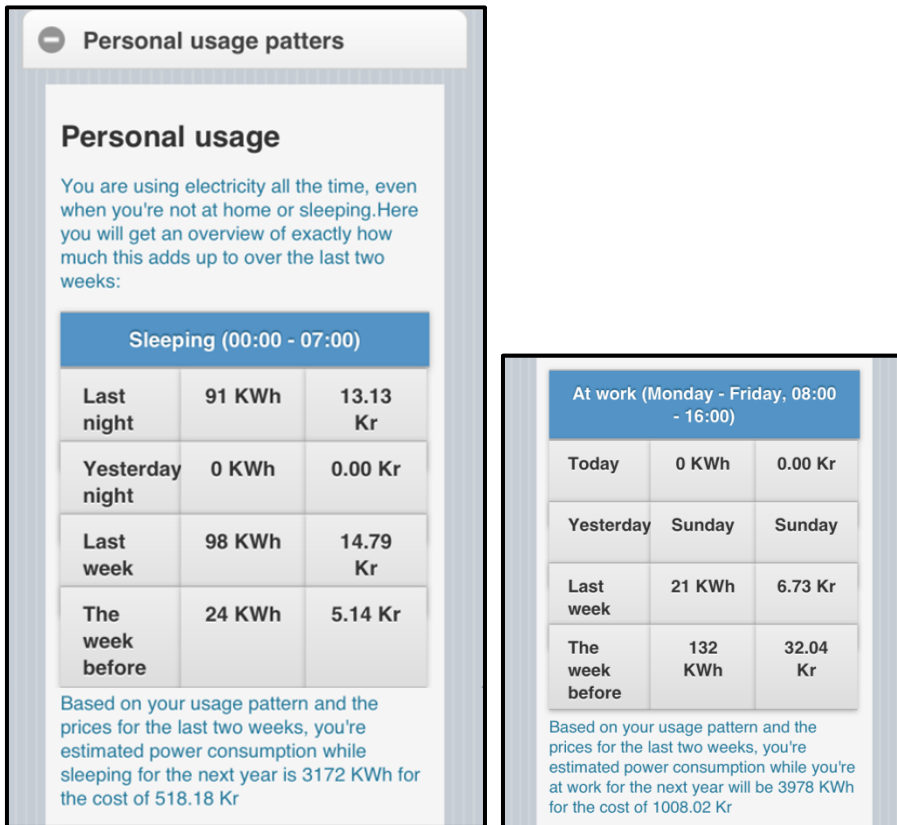


Figure 42: The “Personal usage patterns” can help the user to discover unnecessary power usage

3.4.1.4 Control Page

Due to the fact that there were uncertainties regarding who would supply and install the necessary home control equipment, the feature of controlling devices in the home was left out of the pilot project. However, the device control functionality was developed as a proof-of-concept, discussed in further detail in section 3.4.6.



Figure 43: The Control page

The layout and functionality of the Control page is still planned to be designed as described section 2.2.1.3. The Control page will present the user with the opportunity to turn on/off the appliances in the list as well as controlling the thermostat (if one is present) of the house by setting the indoor temperature, as illustrated in Figure 43.

3.4.1.5 History- and Datepicker Page

On the History page there are in total six graphs with different data, and the first two are shown in Figure 44:

- Correlation between usage and temperature
- Correlation between price and temperature

In addition to these two, the four remaining are as follows:

- Correlation between price and usage
- Correlation between cost and usage
- Correlation between cost and temperature
- Usage in KW

These correlations present the user with useful information about his consumption pattern, thus showing where energy can be saved.

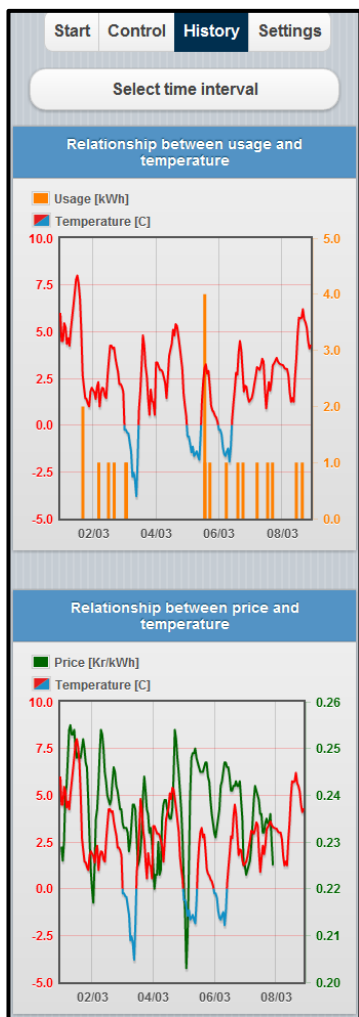


Figure 44: The History page

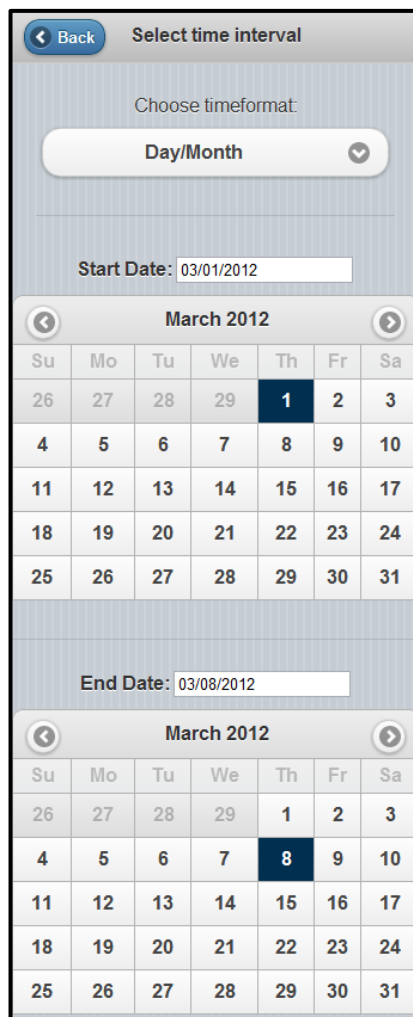


Figure 45: The user can manually set the graphs' time interval

As stated in the requirements, the user should be able to view historic data in the History page. It was decided that it was best to create the six mentioned graphs and let the users select the time interval themselves, as stated in section 2.2.1.3. In that way it is possible to get detailed data for one single day as well as an

overview of the last year, all by simply adjusting the start- and end date of the graphs as shown in Figure 45.

3.4.1.6 Settings Page

The Settings page, illustrated to the left in Figure 46, shows the meter information about the smart meter that is currently the source of data used in the rest of the application. If the user has several meters registered in the database, for example one in the home and one in the cabin, he can switch between these here. Since the Smarties project's pilot customers are living in both Norway and Sweden, it is possible to change the language to Norwegian and Swedish as well as English, shown to the right in Figure 46. However in order to execute the changes, the *Choose meter and go to Start* button must be pressed. That will update the database with new default language and meter, redirect the browser to the Start page, and download the new meter and language data.

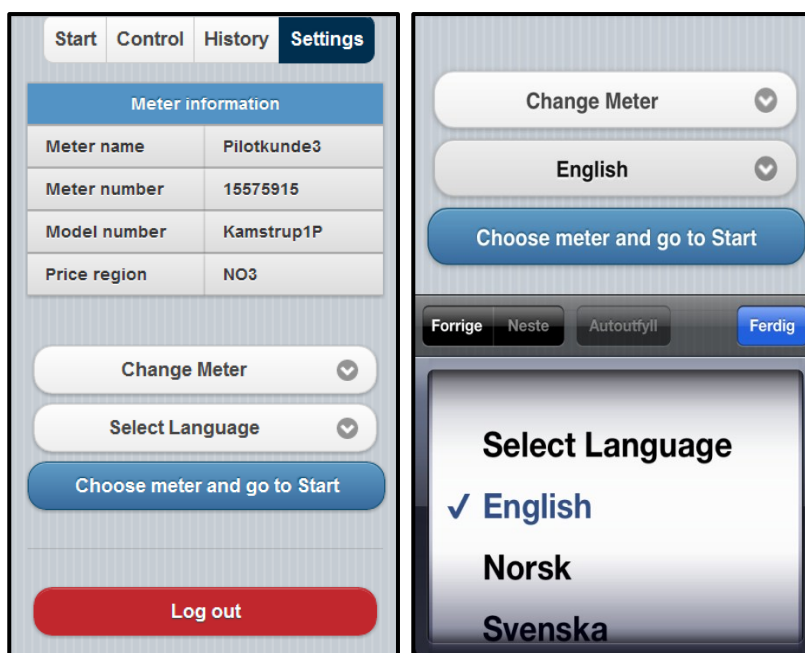


Figure 46: The Settings page

Pressing the *Log out* button will end the current session and redirect the application to the Login page. The functionality of the session handling will be described in section 3.4.3.

3.4.2. Making the Data Available for the Client-Side

With all the data stored in a database, this information now had to be made available for the users. To get an overview of the database, a database diagram was developed, as shown in Figure 47. This model presents the various dependencies and foreign keys used when querying the database.

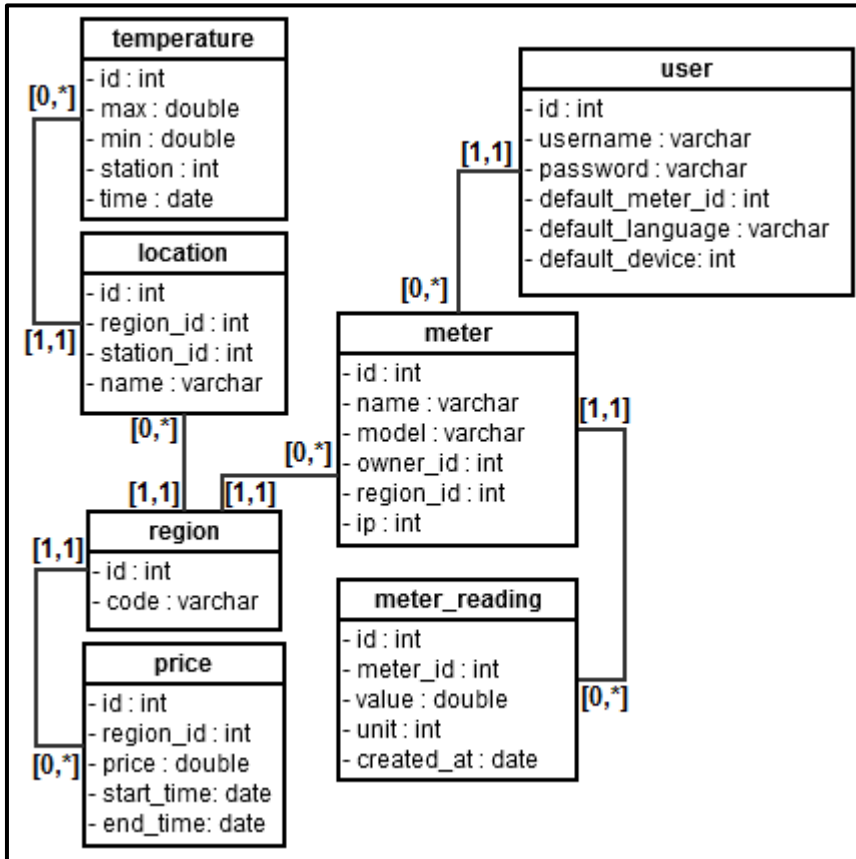


Figure 47: Model showing the relations between different tables in the database

The following example serves as an explanation of the logic in the database model in Figure 47:

Example 3.1

A user can own one or more meters. A meter on the other hand can only have a single owner. The link between the *user* table and the *meter* table is the foreign key *owner_id* in the *meter* table. Likewise, the meter can have zero or more meter readings registered, but a single meter reading can only be associated with a single meter. The link between the *meter* table and the *meter_reading* table is the foreign key *meter_id* in the *meter_reading* table.

By combining this information one can retrieve every meter reading associated with every meter owned by the user in question.

As discussed in section 3.2.2.2 the chosen data-interchange format was JSON. The Pyramid framework makes the task of returning JSON data very easy, but to get an understanding of how it is done, one must also comprehend how Pyramid deals with *views*.


According to McDonough [31], views are bits of code which do something interesting in response to a request made to the application. Its ultimate responsibility is to create a Pyramid response object. To illustrate this, the meter information view is presented in Figure 48.

```
@view_config(route_name='meterInfo',1 renderer='json')2
def meterInfoView(request):
    meter_id = request.params['meterId']

    if authenticated_userid(request):
        return{
            'defMeterId': meter_id,
            'defMeterRegion': Region.getRegionCode(Meter.getRegionId(meter_id)),
            'defMeterModel': Meter.getMeterModel(meter_id),
            'defMeterName': Meter.getMeterNotes(meter_id),
        }
```

Figure 48: A view returning JSON data upon a request

Outline 1 in Figure 48 is the route association for this view. This is the link between the path */meterInfo* and the view. Consider the following URL:

 129.241.104.68:6544/*/meterInfo?*meterId=15575915

By requesting this URL, the view in Figure 48 gets triggered. From outline 2 in Figure 48 one can see how the renderer is set to “jsonp”, thus returning data in the correct format. The returned data is shown in Figure 49. These data is then used by the JS to create the *Meter information* table shown in Figure 46.

```
- {
  "defMeterId": "15575915",
  "defMeterModel": "Kamstrup1P",
  "defMeterRegion": "NO3",
  "defMeterName": "Pilotkunde3"
}
```

Figure 49: The returned JSON data from the */meterInfo* view

The AMS web application makes use of several views and these are presented in Table 2.

Table 2: The various views that make data available to clients

Path	Information provided
<i>/initialData</i>	<p>All the information retrieved upon successful authentication:</p> <ul style="list-style-type: none"> • Current meter reading • Today’s total power consumption • Today’s total cost • Yesterday’s total power consumption • Yesterday’s total cost • Array of price data for current region • Array of temperature data for current region • Array of hourly power consumption data • Array of meter readings for current meter • Statistical consumption data

<i>/xDays</i>	<p>Information returned when user selects a custom time interval:</p> <ul style="list-style-type: none"> • Array of power consumption data (KW) • Array of hourly usage data • Array of hourly cost • Array of temperature data • Array of price data
<i>/meterInfo</i>	<p>Information about the user's smart meter:</p> <ul style="list-style-type: none"> • Identification number • Region • Model • Name

To make more data available for the clients, one can create new views, or add more information to the already existing views. This means that the solution is flexible and will handle future changes.

The JSON data shown in Figure 49 has a flat structure, meaning that it does not contain any objects with underlying elements. However, this is the case with the JSON data returned from the */initialData* view, shown in Figure 50. To access the underlying elements, these must be referenced via the higher-level key(s). E.g. To access the average

```

- {
  - "todaysCost": {
    "todaysCost": "0.00"
  },
  - "price": {
    + "twoDimPrice": [196],
    "max": 180,
    "average": "0.232",
    "min": 175
  },
  "defaultLanguage": "English",
  "logged_in": null,
  - "yesterdaysCost": {
    "yesterdaysCost": "0.00"
  },
  + "temperature": {},
  + "hourlyPowerUsage": {},
  + "totalUsage": [1],
  + "todaysPowerUsage": {},
  + "yesterdaysPowerUsage": {},
  + "meterReading": {},
  + "lastNight": {}
}

```

Figure 50: The JSON data returned from the */initialData* view

price for the next twenty-four hours, the average object simply needs to be extracted from the price array.

3.4.3. Authentication

Most browsers offer the possibility of storing username and password locally in a password manager which makes the login process more efficient and user friendly. However, this practice is a security liability since anyone who can gain access to the device will be able to log in if the password has been stored in the password manager. There is a way of disabling this feature in JavaScript by simply adding an *autocomplete off* attribute to the login form, shown in Figure 51. This will prevent the browser from offering the possibility of saving the password for the current page all together.

```
<script>
    $(document).ready(function() {
        $('#loginForm').attr('autocomplete', 'off');
    });
</script>
```

Figure 51: A simple jQuery entry can disable the password manager for the given page

However, the script presented in Figure 51 has not been included in the final version of the prototype. In discussion with WT it was decided that it was worth the risk to keep the option for storing the credentials in the browser. For the user to see the profit of such an application, daily use would be required since the hourly electricity prices are updated every day. The annoyance of having to enter credentials every time the applications starts could result in users not utilizing the application frequently enough to see the profit of their savings measures, and this could in turn result in them stop using it all together.

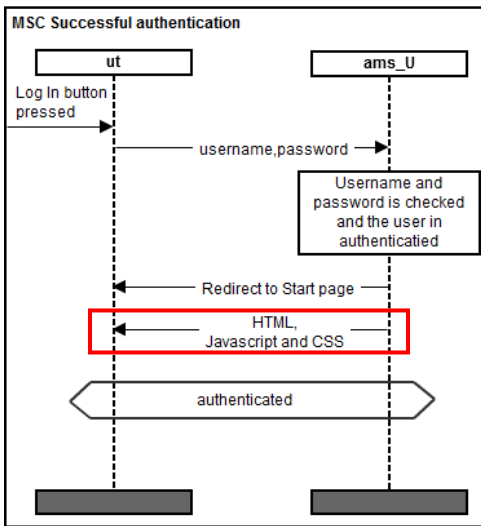


Figure 52: If the credentials are correct, the user gains access to the application

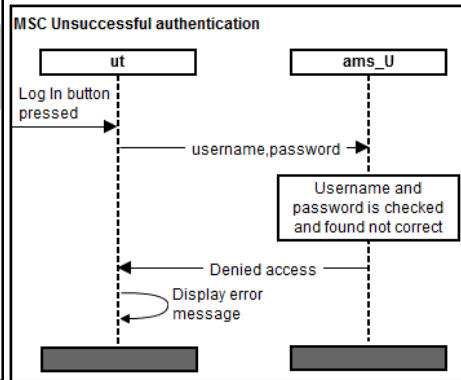


Figure 53: If incorrect credentials are passed to the server, access to the application is denied

Figure 52 and Figure 53 show the communication between the user terminal and the server during the authentication process. Figure 52 shows a successful authentication, while Figure 53 shows an authentication sequence where the user is denied access to the system. When the *Log In* button is pressed, an HTTP request containing the user credentials is sent to the server for authentication. At the server-side, the logic is quite simple as illustrated in Figure 54.

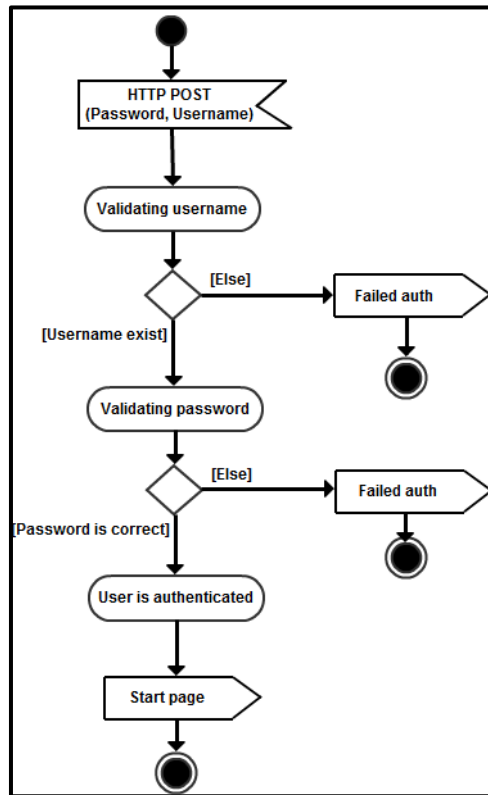


Figure 54: The server-side authentication process

Upon receiving the user credentials, the server executes a database query to check whether or not a user with a username identical to the one received exists in the database. If no such user is found, the requestor is presented with an error message stating that incorrect credentials were passed to the server. If, on the other hand, the username do exist in the database, the authentication process moves on to the next level. The received password is hashed using the Secure Hashing Algorithm-1 (SHA-1) and compared to the password stored in the database⁸. If a match is found, a session is started and the requestor is redirected to the start page. Until the session is ended, either manually or as a result of a time-out, the user's subsequent requests are associated with the session. This means that the server can keep a record of the states of the various authenticated

⁸ All passwords in the database were hashed using SHA-1

users. As will be covered in section 3.5.2.4, sessions also enable user data to be temporarily stored at the server, thus improving the system performance.

3.4.4. Fetch Data from the Server

When a user has been authenticated by the server, the application automatically loads from the server as shown in the outlined section in Figure 52. As the HTML document is interpreted, the browser immediately loads the imported CSS files and executes the JavaScript files in order to style and inject information into the HTML document. The first action in `dataFetchAndDraw.js` when the Start page in the HTML document loads is to perform an AJAX request for the JSON file `initialData`. As stated in section 1.4, AJAX allows client scripts to contact the server for storing/retrieving data without downloading entire web pages, hence resulting in an enhanced user experience due to shorter loading times and more dynamic web pages. An example of how AJAX was used in the AMS web application is shown in Figure 55. This figure shows how the price array containing all the price information is fetched from the JSON document presented in Figure 50.

```
$.ajax({
  url: protocol + "://" + host + ":" + port + "/initialData", 1
  dataType: jsonType,
  crossDomain: false,
  success: function(data) 2
  {
    $.each(data, function(key, val) { 3
      if(key=='price'){ 4
        for(var jsonkey in val){
          if(jsonkey=='twoDimPrice'){ 5
            twoDimArrayPrice = val[jsonkey];
          }
        }
      }
    }
  }
});
```

Figure 55: AJAX is used in `dataFetchAndDraw.js` to fetch JSON data from the server

Explanations to the figure:

1. The AJAX URL is the address where the JSON file is located
2. If the request is successful and the data have been returned, the rest of the code is executed

3. The `$.each()` function runs through every high-level key in the JSON file
4. For every keyword in the JSON file that contains data required by the application, an if-statement executes code associated with the current keyword. In this case it is all the price data that is being handled
5. When a keyword is matched, the data associated with that keyword needs to be extracted and stored appropriately. The example from Figure 56 shows the graph-data for the price information that is stored in an array that will be used to draw the price graph and create the pricelist.

```
1 url: protocol + "://" + host + ":" + port + "/xDays?start="+chosenDates,  
2 url: protocol + "://" + host + ":" + port + "/xDays?dates="+chosenDates,  
3 url: protocol + "://" + host + ":" + port + "/meterInfo?meterId="+meterId,  
4 url: protocol + "://" + host + ":" + port + "/allMeters",
```

Figure 56: The different AJAX requests in dataFetchAndDraw.js file

All data is temporarily stored and used in the application for injecting graphs, tables and other elements into the HTML for it to be displayed to the user. The AJAX example shown in Figure 55 is only one of several server requests. By changing the URL of the AJAX request, different sets of data can be fetched from the server while still using the same function. The remaining four requests are portrayed in Figure 56 serve following purposes:

1. In the history page the users can specify custom time intervals for the various graphs shown in Figure 45. This is the AJAX request that fetches the data for the chosen time interval
2. This URL is used in the AJAX request that is initiated when the Setting page loads. It fetches all the information about the current meter, shown in the *Meter information* table from Figure 46.
3. When a user changes the meter by clicking the *Change Meter* button shown in Figure 46, this request will fetch the *Meter information* for the new meter
4. This request fetches all the meters registered to the authenticated user so they can be listed in the *Change meter* drop down list.

3.4.5. Presenting Data and Drawing Graphs

After the data has been fetched from the server it is ready to be displayed in the HTML document. As mentioned, JavaScript is a dynamic scripting language that can inject data into a pre-existing HTML document without reloading the entire document. With jQuery, it is fairly simple to inject data into the HTML document by using any of the methods presented in Example 3.2.

Example 3.2

In the HTML document, the div tag represents a section of the document. The section ready for injection is `<div id="injectData"> 'code' </div>`. In the JavaScript file there are several commands possible to use for this purpose, the ones used in the `dataFetchAndDraw.js` is:

- `$('#injectData').text("Wanted text to inject");`
 - Replaces everything inside the chosen div with the given text string.
- `$('#injectData').html('Wanted data to inject');`
 - Like `.text()`, but here one can also inject HTML code to be interpreted as code by the browser
- `$('#injectData').append("Wanted data to inject");`
 - Like `.html()`, but here the injected data is simply appended to the present data inside the chosen div section. This option has been used in this project for creating dynamic tables, like the pricelist shown in Figure 40

For the purpose of generating graphs, different JS libraries were evaluated. It was decided to use the plotting library “Flot” [51] which fulfilled the requirements for the JS library offering graph functionality:

- It must be
 - possible to make different types of graphs
 - Lines
 - Points
 - Bars
 - possible to create graph-views with two sharing the x-axis and have different y-axis'
 - possible to choose time format
 - easy to use
 - well documented

```
drawTwoGraphsInOneHistory(priceGraphArray, green,
usageGraphArray, orange, 3);
```

↙ The variables

Figure 57: The function call for drawing graphs in the history page

Figure 58 shows the function in the `dataFetchAndDraw.js` file that creates the two graphs shown in Figure 59. The function is made to be generic, in that way it can be used to create all the graphs in the History page. In this sense “being generic” means that the function has all the necessary input variables needed to create a new graph. When the function is called, showed in Figure 57, all these variables are sent along to the function so that different graphs can be created without having to change anything to the function shown in Figure 59.

```
function drawTwoGraphsInOneHistory(t1,t1Color,t2,t2Color,graphNr){
    options={  xaxis: {mode: "time", timeformat: timeFormatHistoryGraphs},
Graph 1: Usage → yaxis: {position: "left", color: t2Color, margin: 10},
Graph 2: Price → y2axis: {position: "right", color: t1Color, margin: 10},
                legend: {container: $('#historyChartLabels'+graphNr+'')}
    }
    ↗ $.plot($("#historyChart"+graphNr+""), [t1,t2], options );
    }
    ↖ Plots the graph in the chosen <div> section
```

Figure 58: A generic function for graph generation

To show how the function works, the markings in Figure 58 points to which parts of the option is resulting in making the graph to the left in Figure 59. The markings inside the function show the options set for the two y-axis' for Usage and Price, in addition the options for the shared x-axis are set. The option *legend*

creates the two labels above the graph itself, marked in Figure 59. Finally the jQuery command `$.plot(...);` actually creates the graph according to the given options and arrays of data, and injects the graph into the HTML document.

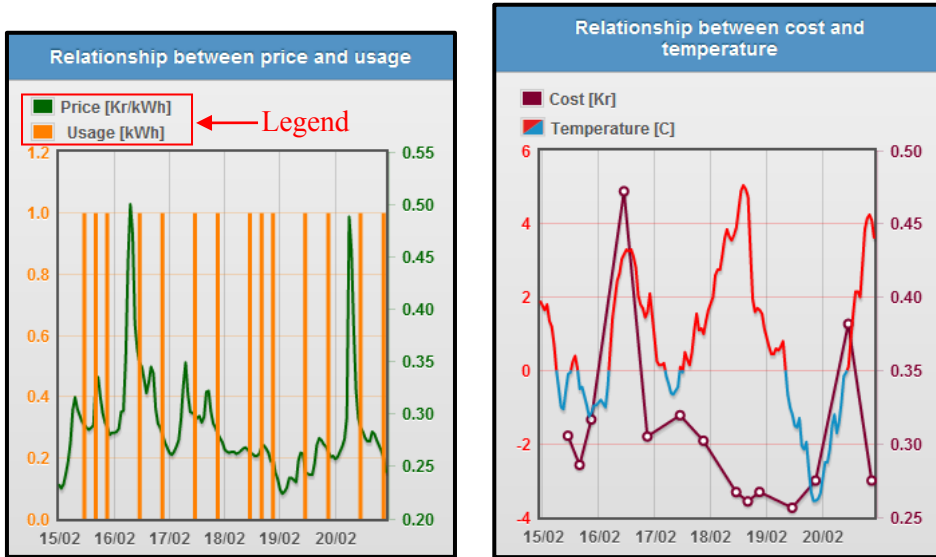


Figure 59: Graphs showing various correlations

If the application is run on a smartphone and the phone is flipped from portrait mode to landscape mode the height and width naturally changes, hence the graphs and tables need to be resized to fit the new window properly. The resize function shown in Figure 60 solves this problem. It triggers whenever the window size of the browser changes, scaling the content of the application to the new window size.

```
$(window).bind('resize',function(){
    //Graphs and tables that needs to
    //resize if the size of the browser
    //window is changed
});
```

Figure 60: The resize function

3.4.6. Controlling Devices in the Home

Due to the change of supplier during the project period, the necessary equipment needed for developing complete device control functionality was not available. The uncertainty regarding who would supply the devices also entailed uncertainty regarding what home control protocol was to be used.

For this reason, the internal home communication was abstracted and the focus was directed towards the control traffic and the internal logic of the server. With the purpose of simulating device control functionality, an Arduino board was acquired. According to the official Arduino web site [52], this is “an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software”. Basically it is a programmable board, as shown in Figure 61.

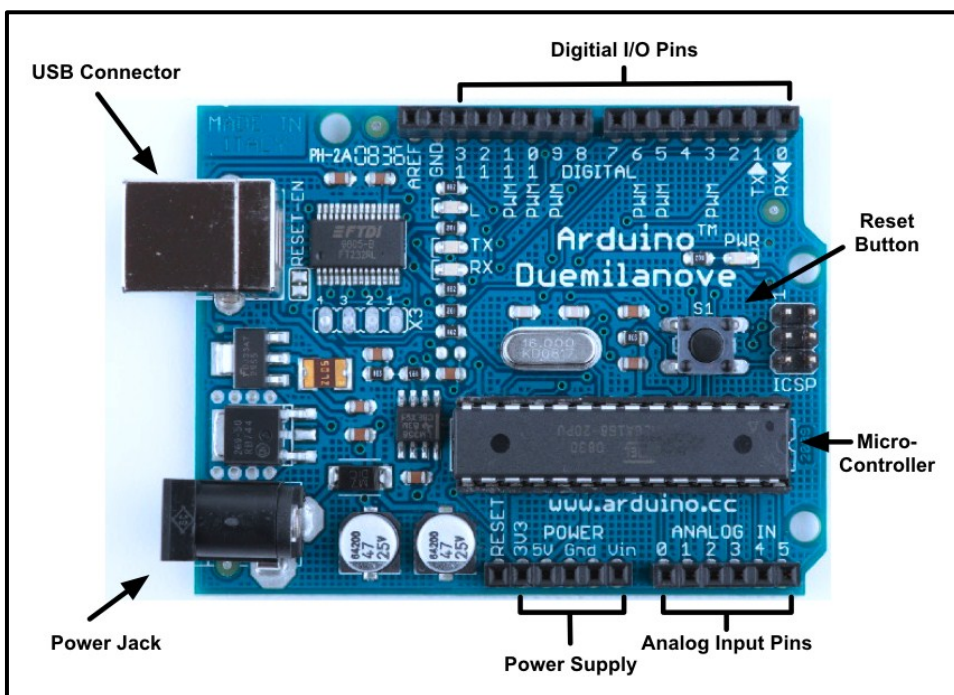


Figure 61: An Arduino board was used to simulate device control functionality [53]

The idea was to simulate a control unit, using the Arduino board. It would handle all communication with the AMS server and serve as an interface to the various controllable devices inside the household. An electronic circuit was built

on a prototyping board to simulate the different devices, visualized in the circuit diagram in Figure 63.



Figure 62: A successful test of the home control proof-of-concept was performed⁹

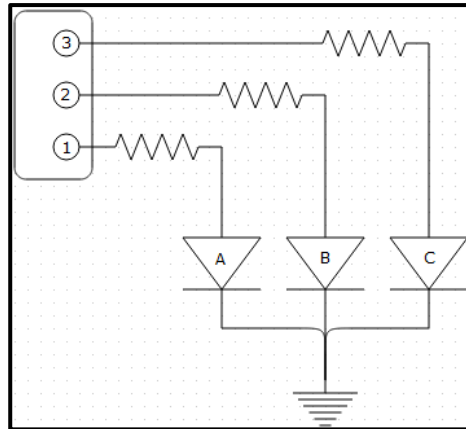


Figure 63: Circuit diagram showing the simulation setup

Each digital pin marked 1-3 in Figure 63 simulates a control module, and each LED marked A-C represents an electrical household component such as a heater, panel oven and so forth. Each anode is connected (via a resistor) to a digital pin on the Arduino board and each cathode is connected to ground.

Using an Ethernet module, the Arduino board is connected to the internet. The software running on the Arduino microcontroller is programmed to listen on a specific port for incoming HTTP requests. Upon receiving an HTTP request, the program will change the state of the digital pins to the state specified in the HTTP request. Thereby, changing the state of a panel oven from OFF to ON in

⁹ The test has been documented on film and can be found on YouTube at this address: <http://www.youtube.com/watch?v=N8DTxVqEtDI>

the application will make the associated LED light up. Figure 62 shows a picture taken during a real-life test of this functionality. One can see that diode A and C is lit up on the prototyping board, while diode B is not. The status of the three buttons on the application’s control page also reflects this.

At the AMS server, database tables for control units and control modules were created to enable access control and to keep track of the states of the various pins, as shown in Figure 64.

- A new column was added to the *user* table. The column *default_cu* holds the control unit ID associated with the user object
- The *control_unit* table holds the Arduino information, such as MAC address, IP address, port and owner
- The *control_module* table holds the state of each of the pins associated with the control unit

The changes to the database mean that unauthorized access to a control unit can be countered. Upon receiving a control request from a client, the server can now query the database to check if the requestor is listed as the owner of the device.

Due to the fact that the state of each digital pin is kept in the database, a user can now terminate the session and later on log in and have the correct states presented in their control page as shown to the left in Figure 43.

Even though the decision on a home control protocol was not yet settled, one had at this stage come up with a working device control-scheme for both server and client. For this scheme to be compatible with

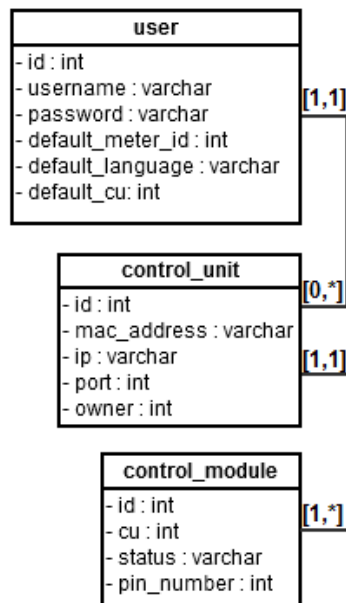


Figure 64: Database tables were created to facilitate device control functionality

special home control protocols, only the communication between the AMS server and the control unit would have to be modified.

3.5. Testing and Optimizing

A system meant to serve a vast group of users must be well planned in order to deal with future expansions. In this section, results from performance tests are presented. Bottlenecks are identified and suggestions on how these can be mitigated are given.

3.5.1. Identifying Bottlenecks

As stated in chapter 2.2, the system development process went through several iterations between the different stages of development in order to achieve the best possible result. One part of these iterations was to continuously test the written code to identify errors and bottlenecks as early as possible and implement measures to correct the problems.

Manual testing was performed by analyzing the network traffic as the page was loaded. As well as the manual analysis, Google Chrome offers an audit feature [54] which gives advice on measures that can improve the web service. These analyses revealed several bottlenecks:

- There were many different JS-, CSS- and HTML files that needed to be downloaded
- The AJAX requests were accountable for most of the loading delay
- Repeated AJAX requests were performed when browsing the application. When loading the Start page, all necessary HTTP and AJAX requests were performed, but if the user browsed to the another page and back to the Start page, the same AJAX requests were performed again, demanding unnecessary resources resulting in unnecessary network traffic.

Since the AJAX requests were identified as the main bottlenecks, the server code was reviewed to pinpoint the areas of improvement. The code review soon showed that the bottleneck was due to the choice of having different user data such as meter readings, price information and cost all be requested separately. Each of the various pieces of information was represented by a unique URL with the same structure as the URL shown in outline 1 in Figure 55, but instead of the path */initialData* the paths presented in Table 3 were used.

Table 3: Unnecessary separation of data caused a bottleneck

Data	Path
Meter readings	<i>/meterReading</i>
Price	<i>/price</i>
Cost	<i>/cost</i>

The code review also resulted in the detection of yet another area of improvement. It showed that some of the paths, like the ones mentioned in Table 3, requested the same data from the database. The *price* path would request price data from the database and return it as JSON data. The same price data would then be retrieved a second time by the *cost* path which would combine price and consumption into cost data.

3.5.2. Optimizing the Application

Based on the test results presented in section 3.5.1, measures were taken in order to optimize the application prior to the pilot period. This section will introduce these optimization measures.

3.5.2.1 Compress the JS and CSS Files

Compressing JS and CSS files in web development is standard procedure for reducing the size of such files to reduce the websites loading time. To perform this compressing, these free services were used:

- JS compression
 - <http://jscompress.com/>
- CSS compression
 - <http://www.csscompressor.com/>

The size of the files can be considerably reduced in most cases. To give an example, the CSS file used in this application was compressed with impressive results:

Compression ratio: 3.842KB/9.889KB = 61.1% (-6047 Bytes)

The compression process is mostly removal of unnecessary bits (backslashes, semi-colons and white space). The compressed code is still readable, however not very easy to understand and modify, therefore the code is written like normal, but prior to release the files are compressed.

3.5.2.2 Application Cache

According to Mark Pilgrim [55], this HTML5 application cache feature, or simply *appcache*, was originally meant to enable web applications to run offline. This is achieved by downloading every file only the first time a website is visited. For subsequent visits, the files are loaded from cache memory. However, the feature can also be used to decrease loading times of an online web application, which is why the AMS web application implements appcache.

```
<html manifest="index.manifest">
  <head>
    <script type="text/javascript">
      window.applicationCache.addEventListener('updateready', function() {
        window.applicationCache.swapCache();
      }, false);
    </script>
```

Figure 65: The code in the HTML file that enables appcache

In order for appcache to function, a manifest file needs to be added in the `<html>` tag, as shown in Figure 65. This file contains the file path of every file that is to be stored in the cache memory. It is also necessary to add an event listener that checks the manifest file every time the page loads. This listener decides if the manifest file has been updated since the last time, and if it has, the

cache is deleted and the files specified in the manifest is downloaded and stored in the cache memory once again. Figure 95 and Figure 96 in Appendix D – *Appcache* show the log of events in a case where the manifest file has been updated and one where it has not been.

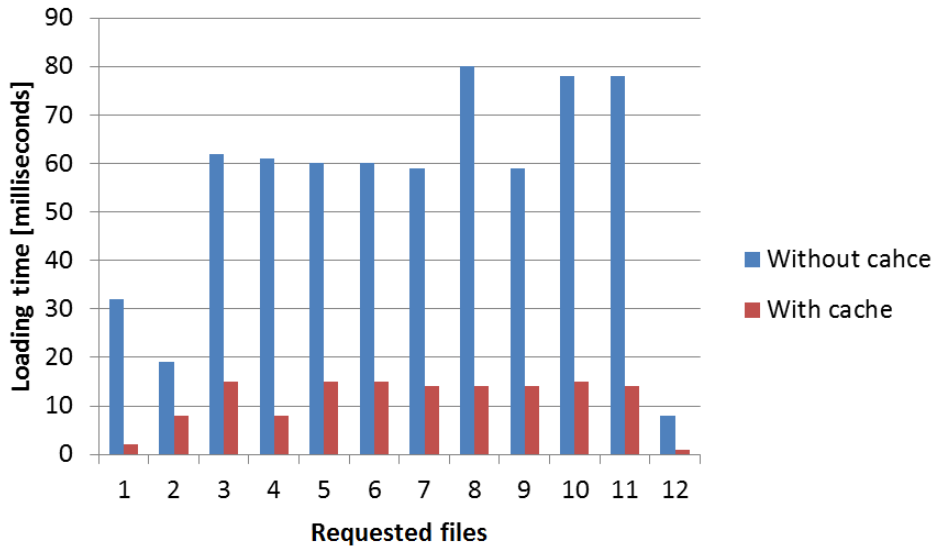


Figure 66: The difference in loading times with and without Appcache is significant

The difference in loading times when appcache is activated is substantial, as shown in the graph presented in Figure 66. This graph has been generated by recording the loading times of the files present in the manifest file. The exact files that are represented can be found in Figure 97 in Appendix D – *Appcache*.

3.5.2.3 Optimize the AJAX Requests

After the initial implementation of the application, it was evident that the AJAX requests needed some adjustments. All of the different data presented in the */initialData* row in Table 2 were originally separate AJAX requests, but after an analysis of the loading-time, it was estimated that the total loading time would be reduced by combining all the data requests into one single request, namely the */initialData*.

However, there was more optimization to be done in relation to the AJAX requests. Initially the data was reloaded every time the Start page loaded, hence resetting the history page graphs if there had been chosen a specific time interval for these graphs. This was fixed by creating separate arrays to store the data for the Start- and the History page. The underlined variables in Figure 67 are the ones storing the Start page data, and these are only updated upon application startup, or whenever the user changes his default meter on the Settings page.

```
var twoDimArrayPrice = [];  
var twoDimArrayPriceStartPage = [];  
var twoDimArrayTemperature = [];  
var twoDimArrayDailyPowerUsage = [];  
var twoDimArrayDailyPowerUsageStartPage = [];  
var twoDimArrayMeasurementKW = [];  
var twoDimArrayMeasurementKWStartPage = [];  
var twoDimArrayHourlyCost = [];
```

Figure 67: The graph data was stored in separate two-dimensional arrays

A positive effect of this separation of data for the different pages is that the graphs in the History page will now remain for that time interval that has been chosen in the Datepicker page, instead of changing back to the last week every time the Start page is visited.

3.5.2.4 Session Data

As a means of reducing the server's workload and lowering the response times, user data were temporarily stored as session data. This meant that the number of database queries could be significantly reduced by only retrieving data from the database for the first request. For every subsequent request, data would be retrieved from the session data. This behavior is illustrated in Figure 68.

After testing the application, the gain from this implementation was clear. The test results showed an average performance gain close to seventy percent compared to before.

It is worth noting that keeping session data may cause problems if load balancing is enabled. Since transport layer load balancers do not distribute the requests according to sessions, it can result in broken sessions according to Microsoft [56]. However, as Microsoft states, there are ways of combining session data and network load balancing successfully.

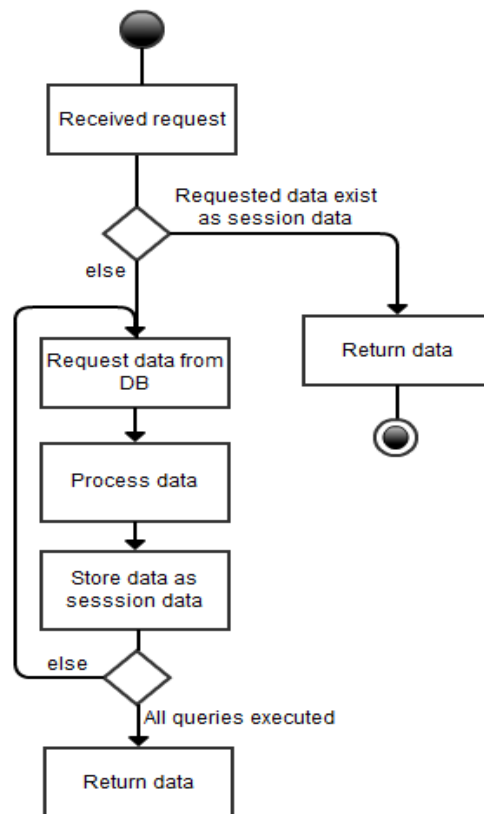


Figure 68: Temporarily storing user data as session data on the server helped reduce the server response times

3.5.3. Performance Testing of the Web Application Prototype

The tool *LoadComplete* [57] by SmartBear was used to create and execute load tests. Reasons for using this tool were, among others, the intuitive graphical interface and the feature for generating custom user scenarios. This enabled realistic scenarios to be automated. Example 3.3 shows such a scenario:

Example 3.3

A user logs in. First, he checks today's price information. Next he browses the control page where he views the status of his devices. Third, he visits the history page and checks his consumption history for the last week. Finally, he logs out, via the settings page.

With different scenarios like the one mentioned in Example 3.3, tests could be executed where the custom scenarios were carried out by *virtual users*, hence mirroring a real-life scenario where a number of concurrent users run the application. By monitoring the test in real-time graph-views, one could see how the system responded to a growing number of users. This graph-view is shown in Figure 69. The light blue graph shows the stepwise increase in virtual users from one to ten, while the dark blue graph is the page loading times.

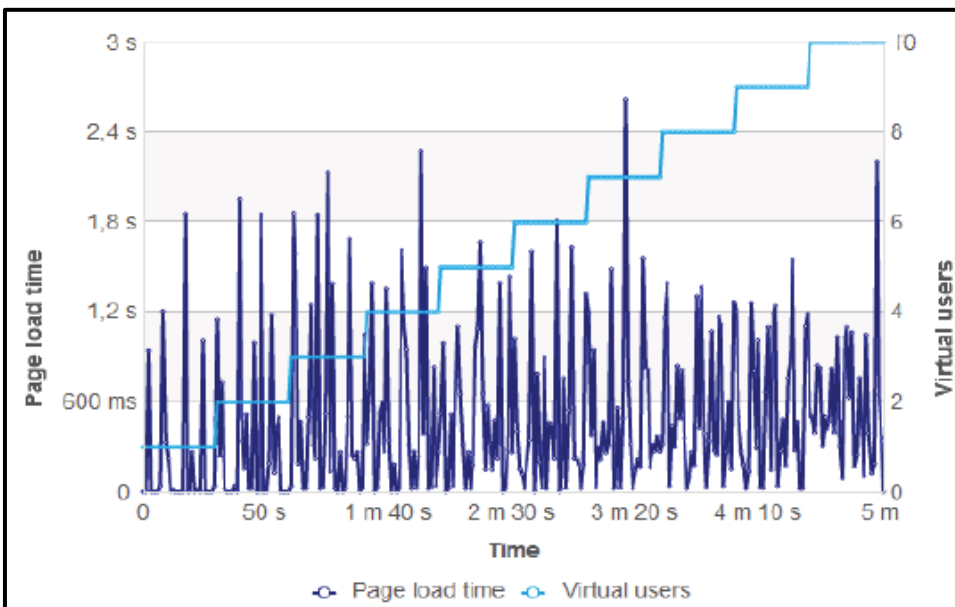


Figure 69: The system can serve up to ten concurrent users without significant problems

As can be seen from Figure 69, the increasing number of users did not cause an increase in page loading times. By monitoring the server using system monitoring tools, we observed how the memory consumption was steadily at 2.9 percent. It did not seem as if ten concurrent users were enough to cause the server any problems.

As a means of identifying possible system bottlenecks, another test was planned. Using the same scenario as the one mentioned in Example 3.1, the number of virtual users was augmented. Starting at ten, the number was stepwise increased by ten until it reached fifty virtual users. The result from this test is shown in Figure 70.

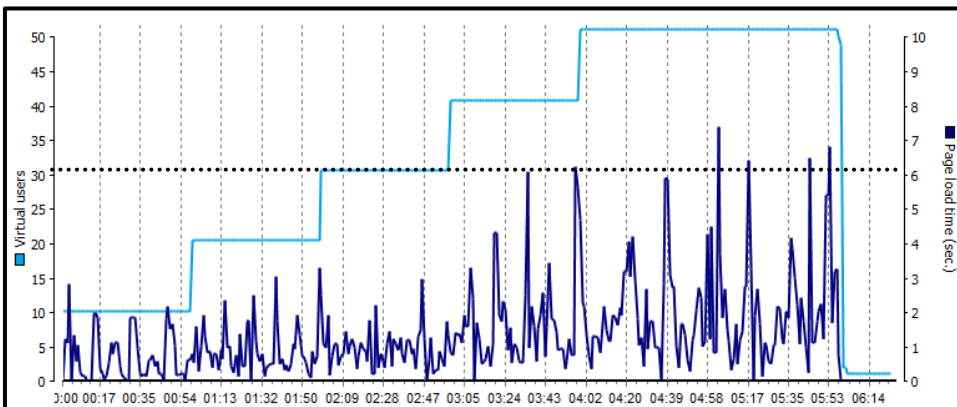


Figure 70: Bottlenecks were discovered as the load was increased from ten to fifty virtual users

From Figure 70, page loading times as high as six seconds can be observed as the number of virtual users reaches forty. We believe that loading times high as these will have negative effects on the user experience.

To identify the bottlenecks, the various requests and responses were examined separately. The most time-consuming responses and their average loading times are presented in Figure 71.

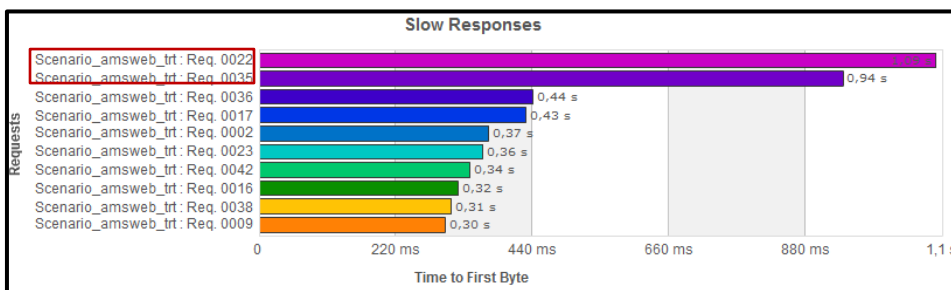


Figure 71: The ten most time-consuming responses were identified

Two responses stand out from the rest:

1. Req. 0022
2. Req. 0035

Request 0022 is the AJAX-request which retrieves the JSON file *initialData*. As explained in section 3.4.2, the *initialData* file contains all user information loaded upon application start-up.

Request 0035 is also an AJAX-request. This request retrieves the JSON file, *xDays*. As shown in Table 2 from section 3.4.2, this request retrieves consumption data from a user-defined period of time. In this test, the time-interval was set to one week.

The transfer time for these requests was as low as ten milliseconds, which meant that the loading time was not due to network limitations. By monitoring the server hardware, one could observe how these two requests were both CPU- and memory consuming server-operations. It was therefore suspected that the cause of the high page loading times were due to hardware limitations at the server. To test this theory, the system was deployed on upgraded hardware. The same tests were performed all over again, and the results of the new tests are shown in Figure 72 and Figure 73.

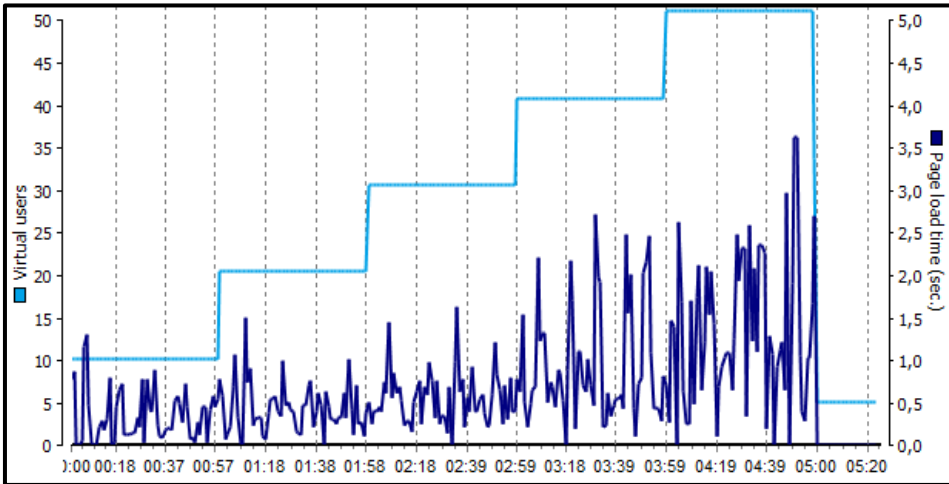


Figure 72: After upgrading the hardware, decrease in loading times was observed

Clearly, the upgraded hardware contributed to shorter loading times. The requests highlighted in Figure 71 were still the most time consuming, but their average execution times were dramatically reduced, as can be seen from Figure 73.

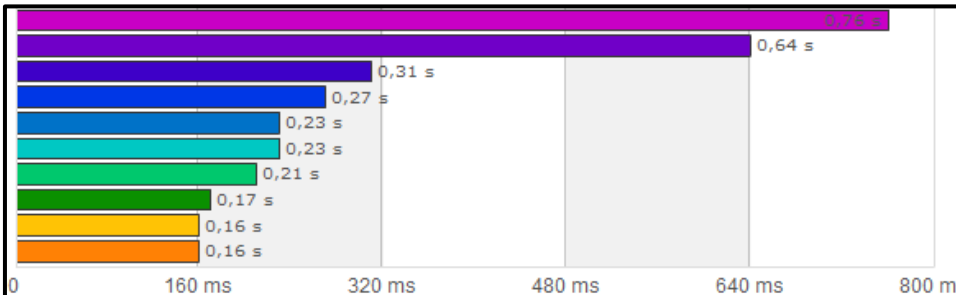


Figure 73: Response times after hardware upgrade

A reduction of over thirty percent for all the requests shows that the presumption was correct. The poor performances discovered in the first test were, to a large extent, hardware related.

4. Security Assessment

In this section the focus is on the different security threats common to web applications. The following assessment is based on the Open Web Application Security Project's (OWASP) Testing Guide v3.0 [58]. As a first step towards an assessment of the security properties of the AMS web application, the different assets important to secure against intruders were identified:

1. The users' personal information
2. The users' controllable devices
3. Passwords

Next, OWASP's list of the most common attacks on web applications was used to develop a risk assessment. A risk assessment is a way of identifying how severe the different threats are. For each attack the probability of its occurrence (P) and the consequence (C) it would entail was evaluated. P and C together made up the risk (R). Each of the three parameters was graded Low (L), Medium (M) or High (H). This logic is explained in Example 4.1.

Example 4.1

An attack X is likely to occur, but the consequence of such an attack is not regarded as severe. An evaluation of such an attack is shown here:

- Probability, $P_X = H$
- Consequence, $C_X = L$
- Risk, $R_X = M^{10}$

¹⁰ Note that this evaluation is based on the tester's judgment and is not the result of a mathematical calculation.

The different attacks and the corresponding risk assessments are shown in Table 4. Threats that represented high risks were highlighted and were subjects to further scrutiny, covered in section 4.1 to 4.7.

Table 4: Assessing the different risks defined in OWASP list of most common attacks.

Entries in OWASP’s list of common attacks	P	C	R
TR 1 - Bypass Authentication			
TR 1.1 – Insecure Direct Object References	H	H	H
TR 1.2 – Insecure Cryptographic Storage	L	H	M
TR 1.3 - Brute Force Attack	L	H	M
TR 1.4 – Broken Authentication and Session Mgt.	M	H	H
TR 1.5 – Insecure Transport Layer Protection	M	H	H
TR 1.6 – Injection Attacks	H	H	H
TR 1.7 – Cross-Site Scripting	M	H	H
TR 1.8 - User Enumeration	H	L	L
TR 1.9 – Path Traversal Attacks	M	H	H
TR 2 – Social Engineering			
TR 2.1 – Cross-Site Request Forgery Attacks	M	H	H
TR 2.2 – Eavesdropping	M	M	M
TR 3 – Server Unreachable			
TR 3.1 – Denial of Service Attack	M	M	M


4.1. TR 1.6 Injection Attacks

Description


SQL injection attacks top the OWASP list of security risks. Reasons for its frequent occurrence are assumed to be its trivial exploitability and its potentially severe impact. An SQL injection attack exploits poorly implemented or non-existing input validation at the server. Such an attack is explained in Example 4.2.

Example 4.2

A web application is offering a service to a vast number of users. The web site has one profile page per user where personal and potentially sensitive data is displayed. When viewing the user profile, the URL looks like this:

 <http://amsweb.trt02.idi.ntnu.no/profile?user=1337>

The attacker alters the user ID in the URL:

 amsweb.trt02.idi.ntnu.no/profile?user=1' or '1'='1

An HTTP request is sent to the server and unfortunately this server does not perform input validation. Upon receiving the request, the server simply extracts the user ID from the request and concatenates it with a predefined SQL query:

```
String Query = "SELECT * FROM user WHERE id='" + request.getParameter('user') + "'";
```

Since the request parameter is simply added to the predefined query, the resulting query is:

```
"SELECT * FROM user WHERE id='" + "'1 or '1'='1" + "'";
```

As a single String this looks like:|

```
SELECT * FROM user WHERE id='1' or '1'='1'
```

Because of the outlined piece of code in the string above, this SQL query is true for all table entries, which in turn means that the attacker will be able to access all users' personal information.

Test step(s)

Inject an SQL statement, like the one mentioned in Example 4.2, in the user input fields at the authentication page.

Test execution

1. SQL statement was entered in the input fields on the Login page as shown in Figure 74.

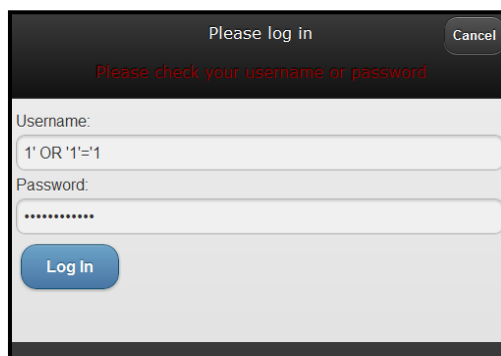


Figure 74: The user input fields on the authentication page were exploited in an attempt to perform an SQL injection attack

2. Continuously monitoring the server logs was an effective way of seeing how these inputs were dealt with at the server side. Figure 75 is an excerpt from the server log which shows how the server encodes the received user input as Unicode. The encoding is one of several features in SQLAlchemy, a Python SQL Toolkit and Object Relational Mapper incorporated in the Pyramid framework.

```
WHERE user.username = %s
LIMIT %s
2012-04-25 16:45:33,483 INFO [sqlalchemy.engine.base.Engine][Dummy-1] (u"1' OR '1'='1", 1)
```

Figure 75: Server logs were used to monitor how the server reacted to an SQL injection attack

- From the log, one could see how the framework encoded the input by surrounding it with double quotation marks:

```
u"1' OR '1'='1"
```

It was therefore interesting to see what would happen if double quotation marks were used at the input as well. New input:

```
1" OR "1"="1"
```

But as Figure 76 shows, this input was also coped with, by surrounding the string in single quotes.

```
WHERE user.username = %s
LIMIT %s
2012-04-25 17:17:23,009 INFO [sqlalchemy.engine.base.Engine][Dummy-4] (u'1" OR "1"="1', 1)
```

Figure 76: An excerpt from the server log after the second injection attack

The result of the framework’s input encoding is an application that neutralizes the special characters often used in SQL injection attacks. Examples of such characters are presented in Table 5.

Table 5: Characters often used in SQL injection attacks

Input character	Meaning in SQL
;	Query delimiter
'	Character data string delimiter
--	Comment delimiter
/* ... */	Comment delimiters

Conclusion

The test did not result in a successful SQL injection attack. The encoding of the user input neutralized the special characters used in the attack. Further, the

authentication scheme provided by the Pyramid framework uses parameterized queries¹¹, in accordance with OWASP's recommendations.

4.2. T-2 Cross-Site Scripting (XSS)

Description

An XSS attack is a type of injection attack, in which malicious scripts are injected into trusted web sites. An attacker performing an XSS attack will use the web application to distribute malicious code, in the form of client side scripts, to a range of other end users. Flaws that allow XSS attacks are somehow related to the injection attacks mentioned in section 4.1 in the way the attacker exploits poorly implemented or non-existing input validation in the web application. The term XSS covers two different attacks:

1. Reflected XSS
2. Stored XSS

A *Reflected XSS* is a non-persistent XSS attack and also the more common of the two. It is not activated by the victim loading the trusted web application, but rather when triggering a malicious URL located on a different site, thus the term *reflected*.

```
<a href="http://ams.idi.ntnu.no?page=<script>/*Something evil*/</script>" />ams.idi.ntnu.no</a>
```

Figure 77: By adding a client side script to a HTML link element, the malicious intent is hidden from the client

To any visiting user, the URL ams.idi.ntnu.no looks like a normal URL, but as shown in Figure 77 it was created with malicious intent. The concept of a reflected XSS attack is presented as an MSC in Figure 78.

¹¹ Parameterizing is a standard procedure for preventing injection attacks by interpreting all user input as plain text and not as SQL code

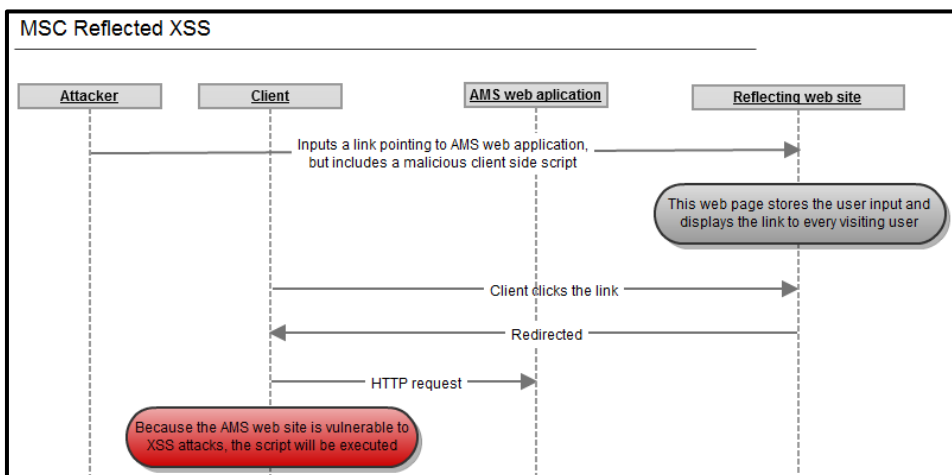


Figure 78: MSC showing the logic behind a reflected XSS attack

A *Stored XSS* is persistent and is triggered by loading the affected web application. By passing a script to the server and having it stored in the database, an attacker may have his script distributed to all the other users of the system. To get an understanding of the logic behind such an attack, a high-level model of a typical stored XSS attack is illustrated in Figure 79.

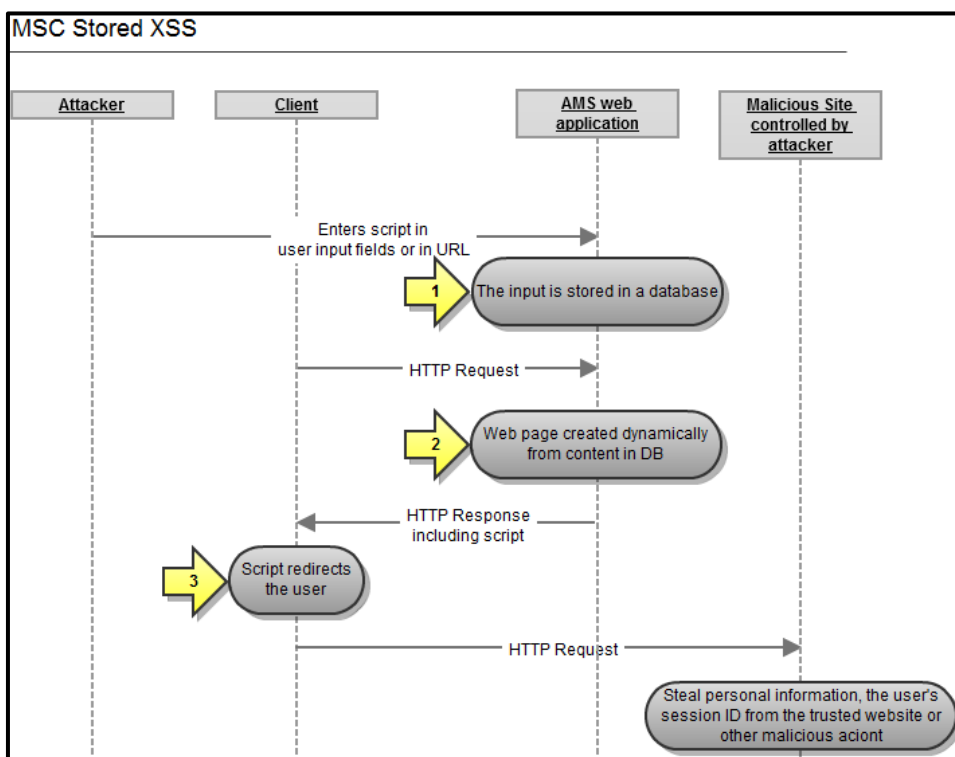


Figure 79: MSC showing the logic behind a stored XSS

Explanations for Figure 79:

1. Upon receiving a user request, the server extracts the request parameters and stores these in the database.
2. As the web page is loaded, the content of the page is gathered from the database. An example is a guestbook site, where earlier posts are retrieved from the database and displayed.
3. Since JavaScript is a client-side¹² scripting language, the downloaded script is executed on the client. By utilizing the possibilities of JavaScript an attacker can retrieve sensitive information from the victim's browser (such as session ID) and forward this information to a malicious site under his control.

¹² Often referred to as *browser-side* scripts

Test step(s)

Locate a URL where a parameter is present. Insert JavaScript code as parameter value

Test execution

1. Located a URL where user input was stored in the database. The URL used for requesting profile language change was chosen.

The URL for switching to Norwegian profile language looked like this:

<http://129.241.104.68:6544/setLanguage?switchToLanguage=Norsk>

One wanted to test whether or not it was possible to insert JS code into the database by altering the *switchToLanguage* variable.

2. The following URL was attempted:

[http://129.241.104.68:6544/setLanguage?switchToLanguage={script>alert\('XSS'\);</script>}](http://129.241.104.68:6544/setLanguage?switchToLanguage={script>alert('XSS');</script>})

3. The database remained unchanged by the attack attempt, hence this type of attack is successfully prevented by the server.

Conclusion

Testing showed that the existing input validation was sufficient. A whitelist, illustrated in the dotted outline of Figure 80, which defines legal options, was used, automatically excluding all other options.

```
@view_config(route_name='setLanguage', renderer='jsonp')
def setLanguageView(request):
    if authenticated_userid(request):
        userObj = User.by_userid(getUserId(request))
        newLanguage = request.params['switchToLanguage']
        if newLanguage in ["English", "Norsk", "Svenska"]:
            userObj.default_language = newLanguage
            DBSession.add(userObj)
            DBSession.flush()
    return None
```

Figure 80: The AMS application uses whitelist filtering when doing input validation. Inputs not present in the whitelist are ignored

4.3. T-3 Broken Authentication and Session Management

Description

In systems where the dataflow may carry sensitive information, there must be ways of authenticating and authorizing users to ensure that this data is accessible only to whom it's intended for, hence the need for an authentication scheme. Web based applications uses HTTP for client-server communication. HTTP is a stateless protocol in the way the web server responds to the client requests separately. Sessions are used to enforce states in a web application, identified by individual tokens called Session IDs or cookies.

There are different ways of doing authentication and session management and often developers design their own customary schemes. Building sound schemes can be hard, thus the possibilities of introducing flaws in different areas, such as:

- Timeouts
- Logout
- Rotation of session IDs

Flaws in areas such as these can result in attacks on the user accounts. If an attack is successful, the attacker gains the privileges of the victim's account and may perform any action the victim is authorized for.

Test step(s)

1. Check whether or not session IDs are rotated
2. Check whether or not sessions time out

Test execution

In order to check for rotation of session IDs, the following test was performed several times:

1. Capture traffic to and from the client using Wireshark¹³
2. Authenticate a user
3. Retrieved the session ID by inspecting TCP packets in Wireshark
4. Terminate user session
5. Re-authenticate the same user
6. Retrieve the session ID a second time
7. Compare the two session IDs

The rotation of session IDs seemed to be OK and satisfyingly random. To test whether or not the application automatically times out sessions, the session was allowed to live for the period of time specified as the session's time to live (TTL). The page was then reloaded, and it was observed how the user was no longer authenticated.

Conclusion

No vulnerabilities were found in the authentication and session management. During the development of the AMS web application, established and proven authentication- and authorization- and session schemes were used.

Combined, these modules helped reduce the risk of design flaws causing security vulnerabilities in the authentication and session management.

4.4. T-4 Insecure Direct Object References

Description


Although users are authenticated and authorized before being granted access to a system, the potential threat from system users with malicious intent must be

¹³ A packet analyzing tool


considered. An authenticated user must not be authorized to access another user's data. This issue is illustrated in Example 4.3:

Example 4.3

Bob is a user of the AMS web application and his username is *bob*. He enters his user credentials in the login form and is authenticated and redirected to his profile page. The URL in the address bar looks like this:

 <https://ams.idi.ntnu.no/profile?userId=bob>

Bob knows his friend Alice is also a user of this service and he correctly guesses her username to be *alice*. He edits the URL by changing the username parameter:

 <https://ams.idi.ntnu.no/profile?userId=alice>

Since the application is vulnerable to insecure direct object references, Bob is authorized to access Alice's personal information.

As shown in the example, a system vulnerable to insecure direct object references may be subject to a substantial data leakage problem, and the seriousness of the problem depends on the data being leaked.

Test execution

All the accessible web pages within the application were browsed in order to check whether or not the different URLs contained variables that could be exploited. None of the pages did contain any URL variables, but as the users change their default meter, an AJAX request is sent to the server with the following URL:

 <https://ams.idi.ntnu.no/setMeter?switchToMeter=15575915>

The URL variable outlined above is the identification number of the meter. A test was performed to check if it was possible to gain access to another user's smart meter by changing the identification number.

Conclusion

No insecure direct object references were found in the application. The attempt to access another user's smart meter did not succeed. The reason for this is the server's validation of the request variable, as shown in Figure 81.

```
@view_config(route_name='setMeter', renderer='jsonp')
def setMeterView(request):

    try:
        {newMeter = int(request.params['switchToMeter']) 1}

        if authenticated_userid(request):
            userObj = User.by_userid(getUserId(request))

            {if userObj.id == Meter.getMeterOwner(newMeter): 2}
            userObj.default_meter_id = newMeter
            DBSession.add(userObj)
            DBSession.flush()

        return None

    except:
        return None
```

Figure 81: User requests with the intention of changing a user's default meter ID were properly handled at the server side

The two outlined sections in Figure 81 are explained below:

1. Inside a *try statement*¹⁴ block, the request variable is retrieved and parsed to an integer. If for some reason the input is not a number, the application will raise an exception and skip the remaining code.
2. The database is queried in order to check whether or not the requesting user is the owner of the meter with the ID number submitted as a request variable.

¹⁴ Try/catch statements are used in software development to catch exceptions where something unforeseen happens. If something fails inside a try block, the code interpreter will simply skip the remaining try block code and jump to the corresponding catch (or except) block


4.5. T-5 Cross-Site Request Forgery (CSRF)

Description

Cross-Site Request Forgery is, as the name implies, forged HTML requests created by an attacker to exploit users that are logged in to an online service. The nature behind such an attack is illustrated in Example 4.4.

Example 4.4

Thomas is logged into his online banking service:

 <https://example.bank.com>

- While still in a bank session, Thomas opens another tab in the browser and visits a malicious site
- The malicious site contains an element generating requests to the trusted site. The request is hidden inside an image tag like this:

```

```

- The malicious link is camouflaged inside an image tag, where both height and width are set to zero, thus not visible on the web page
- Because the victim is logged into the trusted bank service and since the web application is vulnerable to CSRFs, money will be transferred from the victims account to the attackers account without the victim's awareness

Test step(s)

Use a malicious link to try to change a user's default meter. Hide the link inside an image tag to avoid the victim's attention.

Test execution

1. A simple HTML document was created on a different web server:

```
<html>
<body>
<p>Welcome</p>

</body>
</html>
```

2. A user was logged in to the AMS web application

3. The malicious site was visited

4. Updating the AMS web application

When the application was updated in test step four, the user's default meter had been changed.

Conclusion

The test showed that the application is vulnerable to CSRF attacks. A way to fix this issue is to introduce a random token in the HTTP requests:

```
<input type="hidden" id="_req" name="_req" value="your secret key" />
```

By adding a field such as this in the HTML document the token can be stored for subsequent requests:

- The *type="hidden"* will prevent it from being shown openly when browsing the site.
- The *value* field will hold the random token.

By adding the token to the list of parameters sent to the server, as shown in the outline of Figure 82, the server can validate the origin of the incoming requests.

```

```

Figure 82: By including a random token in the HTTP requests, CSRF attacks are counteracted

Since a CSRF attack originates from a different web page, it will not be able to access this token and consequently the forged requests will be discarded once it is received by the server.

4.6. T-6 Path Traversal Attacks

Description

Unauthorized access to privileged pages may occur when a user's behavior pattern differs from the one predicted by the developer. By manually editing the URL, an attacker might access files he otherwise wouldn't have been able to access.

Test step(s)

Try accessing a file outside the application root folder, see outline 1 in Figure 83. For the purpose of path traversal, a technique called *dot-dot-slash* (`../`) will be used. The dot-dot-slash tells the browser to move one level up in the folder hierarchy.

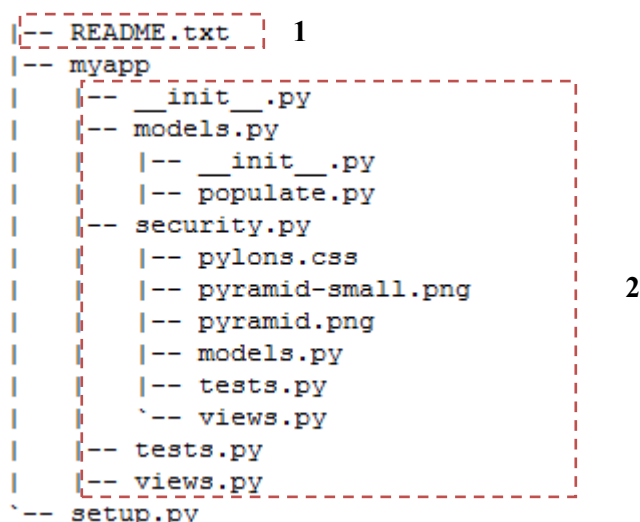



Figure 83: It was attempted to access a text file (outline 1) outside the application's root folder (outline 2)


Test execution

The application was started and path traversal attacks were attempted:

1. It was attempted to access a file outside the application by altering the URL, as shown here:

 <http://ams.idi.ntnu.no/../../README.txt>

The application encoded this URL as shown under, returning a 404 Page Not Found response.

 <http://ams.idi.ntnu.no/README.txt>

2. Several encodings for dot-dot-slash (../) were attempted:
 - %2e%2e%2f
 - %2e%2e/
 - ../%2f

However, none of these encodings could successfully access the file.

Conclusion

These results does not guarantee that the application is resistant to path traversal attacks, but it verifies that it is protected against the most frequently used method of performing such an attack.

4.7. T-7 Insufficient Transport Layer Protection

Description

If nothing is done to secure the data transmission, anyone monitoring traffic between endpoints might acquire valuable or sensitive information. Consider a web application using a login scheme, but no transport layer protection. If traffic is monitored during authentication, an attacker might acquire user credentials as clear text.

Common causes for insufficient transport layer protection are:

- Not using SSL/TLS at all
- Use of SSL/TLS only during authentication, thus exposing the session ID to interception once the user is authenticated.
- Certificates are also a common source of error. If these are improperly configured or if they are expired, they should not be accepted by the system.

Test step(s)

Capture and analyze traffic to and from the AMS web application server.

Test execution

By using Wireshark, each packet could be analyzed separately as it was sent or received. Encryption was at this stage not enabled and it was desirable to check whether or not it was possible to retrieve valuable information from the captured packets. As shown in Figure 84, the user credentials were easily extracted from the captured TCP packets. This observation confirmed the need for transport layer protection during authentication.

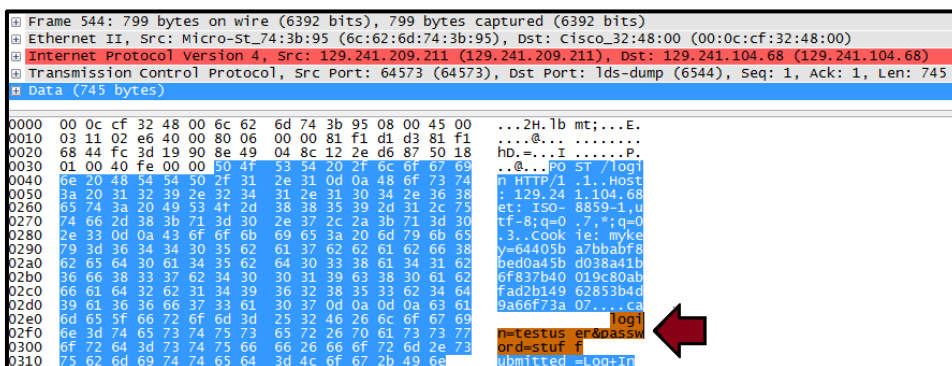


Figure 84: Wireshark was used to capture unencrypted username and password

When a user is successfully authenticated, the server will return the session ID that authenticates the subsequent user requests. As with the transfer of user credentials, this is important to hide from outsiders as it can be used to hijack a session. As suspected, the session ID was also easily extractable from the captured TCP packets, as shown in Figure 85.

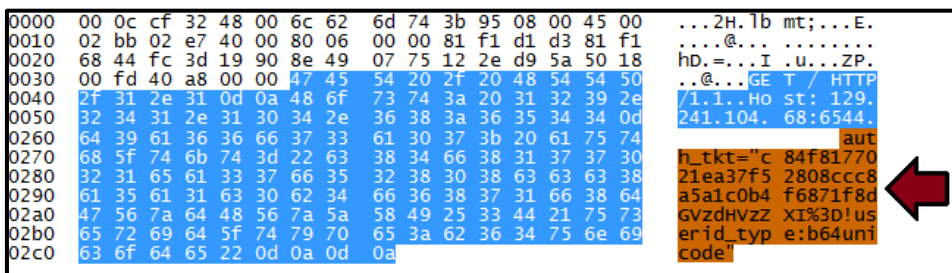


Figure 85: Wireshark was used to capture session ID

To confirm that exposed session IDs are exploitable by someone with malicious intent, a session hijacking was attempted.

Creating and editing cookies are trivial tasks in most browsers. By using tools such as Cookies Manager+ in Firefox, users can create their own cookies and enter values retrieved using traffic sniffing tools. An example where the stolen session cookie is added to the attacker's browser with the purpose of hijacking a session is shown in Figure 86.

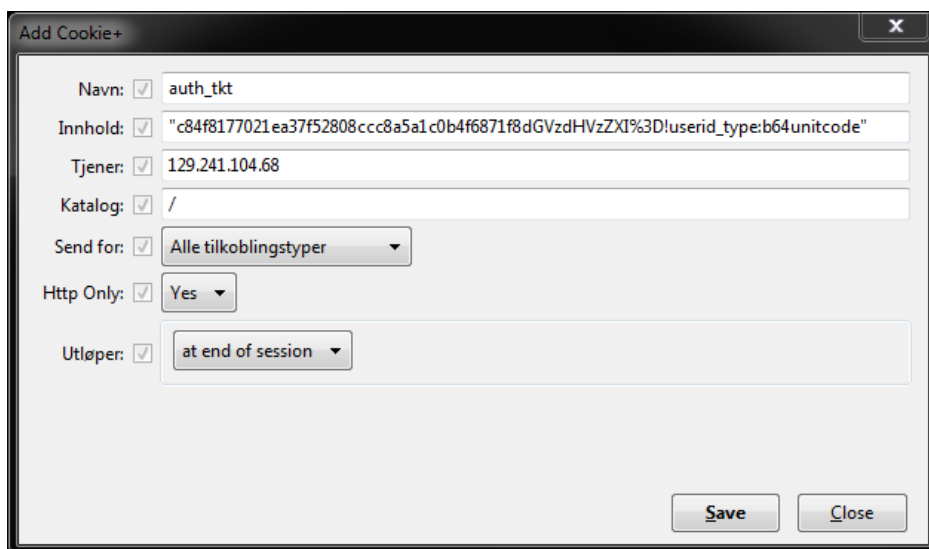


Figure 86: Tools such as Cookies Manager+ makes it easy for anyone to edit and add cookies

After adding the cookie to the attacker's browser, the browser was directed to the AMS web application. The session was successfully hijacked, hence underlining the importance of encryption for authenticated server requests.

Conclusion

As discovered during the assessment, the need for encryption was evident. SSL/TLS was implemented to counter for the security breaches discovered. A new packet analysis similar to the one shown in Figure 84 was performed after this implementation. The result is presented in Figure 87. As shown in the highlighted area, the content is now incomprehensible compared to what we saw as clear-text before.

```

[Calculated window size: 9152]
[Window size scaling factor: 64]
▶ Checksum: 0x4975 [validation disabled]
▶ Options: (12 bytes)
▶ [SEQ/ACK analysis]
▼ Secure Sockets Layer
  ▼ TLSv1 Record Layer: Application Data Protocol: http
    Content Type: Application Data (23)
    Version: TLS 1.0 (0x0301)
    Length: 736
    Encrypted Application Data: b16a2df9b0e2484f202c87bc97905f479620737c6ac3aacc...
  ▼ TLSv1 Record Layer: Application Data Protocol: http
0040 b5 eb 17 03 01 02 e0 b1 6a 2d f9 b0 e2 48 4f 20 ..... j-...HO
0050 2c 87 bc 97 90 5f 47 96 20 73 7c 6a c3 aa cc d1 .....G. s]j...
0060 c1 38 20 47 66 3e 61 8a ff 4f bc f3 11 a7 3b 2f ...8 Gf>a. .0....;/
0070 5f 5c be 66 66 16 27 50 c9 9a 8e 68 c7 0f 3d fe ...\.ff."P ...h..=.
0080 97 7c 75 9e 74 d0 15 1f 6c 5b 54 60 5c 5a 1c 2d ...|u.t... \[T`Z.-
0090 5a 17 98 36 99 7f f2 b1 29 6a 13 6e bf 22 63 31 ...Z..6.... )j.n."c1
00a0 88 cd 08 74 eb ca 56 2d b7 62 34 a3 c9 d3 07 41 ...t..V- .b4....A
00b0 63 7d 6e e3 44 ff 4c a4 83 27 cf c0 04 a6 90 11 ...c}n.D.L. ....
00c0 ac b1 db 92 9e 7a 47 a3 77 18 5b f9 f4 15 dc 60 ...zG. w. \....
00d0 ee 1f 84 a2 c5 d5 fb 4e b5 6d 43 5d 3e da 28 e2 .....N .mC>.(.
00e0 15 f4 15 8c 27 ce c5 07 fc 7a 81 18 16 0c c5 1d .....z....
00f0 b5 c7 a6 99 33 82 95 ce 5e 44 cd 4b 0f 2f 4c 43 ...3... ^D.K./LC
0100 bd 6b a9 a6 53 92 c5 ef 77 70 12 5c 09 f1 01 bd ...k..S... wp.\....
0110 e9 e4 51 55 b7 e3 e8 c9 3a ab d8 8f 66 06 7a a3 ...QU.... .f.z.
0120 17 ec 3e b3 90 87 40 54 44 fc 5f 19 f8 2b 3e 07 ...>...@T D. ...+>.
0130 15 ad 7f cc 24 f1 37 86 20 d1 c1 69 da d1 bf 23 ...$.7. .i...#
0140 23 77 e7 9e 7a 0b 0d 43 57 89 89 4a b6 79 ed 62 #w...z..C W..J.y.b
  
```



Figure 87: When SSL is enabled, the data is encrypted and is therefore incomprehensible to anyone monitoring the traffic

The server certificate used in this project was self-signed because one did not have access to certificates signed by a trusted certificate authority¹⁵. For this application to be used commercially, it is recommended to acquire such a certificate.

¹⁵ Examples of Certificate Authorities are VeriSign, GoDaddy and Comodo

5. Discussion and Future Work

Since the AMS web application development is part of a pilot project, the features are suggestions to how the various requirements can be realized. The current system is a working prototype designed for probing how such a system will be perceived by potential users. However, since both the architecture and the implementation is thoroughly documented and designed with a focus on being easily expandable, a solid foundation for further development is laid.

In this chapter, a critical analysis of the different design- and implementation decisions will be presented. It will address the various parts of the system and identify areas where more work is needed. Further, several alternative solutions for improving the system will be discussed, as well as ideas for new functionalities for future versions.

5.1. Improvements to the Data-Collection

In section 3.5.2, different measures taken to optimize the application were discussed. However, there are further actions that can be done to further improve the performance. In this section both ongoing and future work in regards to improving the data collection part of the Smarties system will be discussed.

5.1.1. Fetch Data from the Meter Logs

For the Smarties system to be able to guarantee accurate consumption data, one very important feature needs to be implemented in the data-collection process. The smart meters are set to initiate connection with the server on a certain pre-set interval, but because these connection attempts may fail, measures need to be taken to collect the missing data from the smart meter's log file. This is a likely

scenario since errors might occur on both the server and the smart meter. In addition, the smart meter may lose internet connection because of errors in the user's home network. At this stage of the project, there are also problems with the smart meters that result in periods where no data is received, as illustrated in Figure 88. If this system is to be used in cooperation with the power companies to give them consumption data for the calculation of electrical bills, this is an important issue to resolve. If left unresolved, this problem leaves the application open to be exploited by dishonest users to unplug their internet connection during periods of high prices, and reconnect it when the prices are low in order to save money. In addition, the data presented to the user is incorrect when errors occur, as Figure 88 clearly shows.

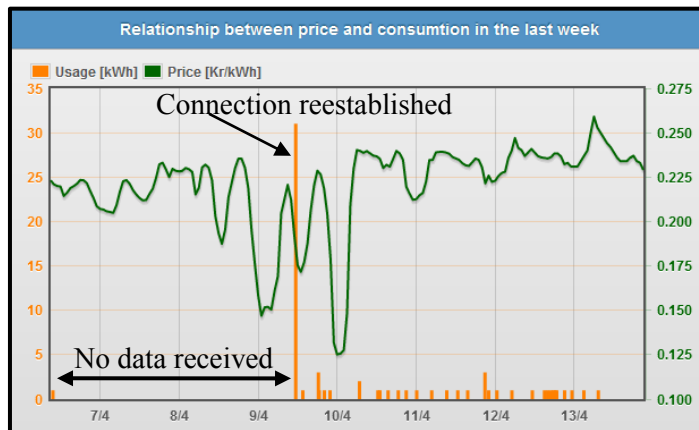


Figure 88: When connection is lost between the smart meter and the server, the data presented to the user is faulty for the down period

There is ongoing work being done to fix this problem, but the new meter data retrieval process is not yet finished. The idea is to have a separate process that will initiate a request for the missing data when connections are reestablished after an error. This new process will request all the missing data polls, and set the timestamp of the measurement to the time it was registered in the log instead of the time it was stored in the database. This would eliminate the false results showing very high consumption values at the time the connection is reestablished, as shown in Figure 88.

5.1.2. Minimizing the KWh Collection

Since the prices only change every hour, there is no need for registering KWh values more than once every hour, at *hh:59:59*, allowing the application to calculate the usage for the preceding hour. The time interval for sending the UDP Hello packets that initiates the meter retrieval process is by default every five minutes on the Kamstrup meters. If that was changed to once every hour, the benefits are clear according to these calculations:

$$\text{Every 5 minutes: } \frac{60 \text{ min}}{5 \text{ min}} = 12 \text{ requests/hour}$$

$$\text{Once every hour: } 100\% - \left(\frac{1}{12} \times 100\%\right) = 91.67\%$$

Implementing this change in the smart meters would consequently decrease the KWh part of the database with astonishing 91.67 percent, as well as decreasing the network traffic correspondingly.

Example 5.1 shows how much the hypothetical reduction of requests in a setting where the user base has grown to consist of 100 000 smart meters.

Example 5.1

requests today – possible requests = # of requests saved

$100\,000 \times 12 - 100\,000 = 1.1 \text{ million requests}$

With a user base of 100 000 registered meters, this change would reduce the amount of requests with 1.1 million every hour, thus reducing the storage space used for meter readings accordingly.

In addition freeing up storage space and reducing network load, this would result in more readable graph presentations.

5.1.3. More Tables in the Database

As mentioned in section 3.5, the main bottleneck on the server is the process of requesting the data from the database and generating the JSON responses. The task of retrieving and calculating the cost is the most time consuming since it requires a combination of several relatively heavy operations:

1. The application retrieves all the consumption data from the database within the chosen time period
2. The application retrieves price data for the same time period
3. Every consumption entry is multiplied with the associated price entry
4. To get the total cost of the chosen time period, every cost entry is summarized

A possible way of reducing the workload per user request is to generate a new database table containing pre-calculated cost values. Every day, the total usage of the given- and the previous day is calculated in both KWh and NOK. This information is displayed in the usage table on the Start page, presented in Figure 39. By storing this information in the aforementioned table, this enables the calculations to only be performed once every day for each user. Subsequent requests will then simply extract this information from the new database table, illustrated in Table 6.

Table 6: By creating a table with pre-calculated cost values, the loading time per request can be reduced

User	This week	Last week	This month	Last month	This year
1	NOK	NOK	NOK	NOK	NOK
2	NOK	NOK	NOK	NOK	NOK
3	NOK	NOK	NOK	NOK	NOK

This solution will decrease the processing time per user requests, but as the user base grows, this solution can cause unnecessary server workload. These calculations would be futile for every inactive user, hence wasting processing time which could otherwise be used to serve active users.

5.2. Usage and Cost Comparison on the Start Page

The usage table on the Start page can be used to show more information. A useful new feature would be to have two new entries showing consumption and cost so far this month, as well as for the same time last month, shown in the outlined section of Figure 89. An example of this functionality is given below:

Example 5.2

If today was the 20th of May, the *This month*-entry would be comprised of calculated data for the first twenty days of May, and the *Last month*-entry would contain data for the first twenty days of April.

With this feature the user could compare the consumption month by month and at the same time be able to see if any potential power saving efforts has been successful.

Start	Control	History	Settings
Usage			
Today	7 KWh	2.75 Kr	
Yesterday	10 KWh	3.46 Kr	
This month	326 KWh	153.13 Kr	
Last month	447 KWh	195.47 Kr	
Total usage		773 KWh	

Figure 89: A suggestion on how the monthly comparison feature could look like

This feature could easily be realized by reusing code from the existing *xDays* functionality that calculates the cost within a certain time period. The *xDays* functionality is used to retrieve data for a time period specified using the datepicker presented in section 3.4.1.5. As presented in Table 2 of section 3.4.2,

the *xDays* response contains five arrays of data, but for this purpose only the power consumption- and cost arrays are of interest. By requesting these two arrays for both time periods, the result would be as showed in Figure 89.

5.3. Upgrading the Pricelist Page

The Pricelist page contains the table with the hourly prices for the next day, as illustrated earlier in Figure 40. This list is updated daily at 15:00, replacing the current list with the data for the next day. Obviously this is not the most optimal solution, since after 15:00 the users no longer have access to the price information for the rest of that given day. A better solution is illustrated in Figure 90 where the rest of the current day is displayed under the header *Today* and then after midnight the list for the next day is presented in the same way as today.

Time	Price
Today	
21:00 - 22:00	0.131
22:00 - 23:00	0.123
23:00 - 24:00	0.127
Tomorrow	
00:00 - 01:00	0.135
01:00 - 02:00	0.125
02:00 - 03:00	0.134
03:00 - 04:00	0.155
04:00 - 05:00	0.183
05:00 - 06:00	0.179

Figure 90: A suggestion on how to improve the pricelist feature

This upgrade would not demand much new code either, since the current function that fetches the content for the table only needs to be expanded to also include the remaining hours of the current day in the same query to the database. The maximum- and minimum values for the two days would still be held separate, so that the list could show a minimum value in the *Today* list as well as

in the *Tomorrow* list if that was the case. Figure 92 illustrates a case where this occurs.

5.4. Enhanced Power Saving Tips

The power saving tips presented on the Start page are only available in English. Naturally these must also be translated to other language, or maybe even rewritten to fit the climate of the country in question. For instance, power saving tips would be different for the United States where air conditioning is one of the most power consuming appliances, as supposed to Norway where heating uses the most electricity in a normal household.

The *Personal usage patterns* part of the power saving tips, explained in section 3.4.1.3 utilizes the user's own consumption data to show areas where there are potential for saving electricity. As this is a proof-of-concept solution, there are several additional enhancements that would improve this feature. The pre-existing tables showing the usage and cost during the night and work hours have hard-coded timeframes, meaning that it does not necessary fit with every user's work- and sleep patterns. To improve this, the user needs to be able to change the settings of his individual table:

- Change the start- and end time for when the user is away for work
- Change the start- and end time for when the user is sleeping
- Set start- and end time for when the user is on vacations

One way to make this feature even more accurate is to let the user register the exact time he leaves his home as well as when he returns. However, it is too much to ask of a user to remember this registration twice every day. A solution to this problem could be to utilize the wireless control unit installed in the house. If everyone in the household is equipped with a wireless tracker, for instance attached to their keychain, the system could automatically register when the house is empty, as well as creating statistics of how much electricity is consumed when the different people are home. Really accurate and helpful

power saving tips could be presented to the user if this scenario became a reality. This tracker-functionality will also be discussed further in section 5.5.

More saving tips opportunities might present themselves in the future, depending on the functionality of the control modules, shown in Figure 2. If they provide consumption information about the associated controllable device, algorithms can be developed to generate suggestions of when certain appliances can be turned off in order to increase the power efficiency, as well as estimating the impact of such a measure.

5.5. Control Devices in the Home

Since the control functionality could not be accomplished as planned because of the problems with the supplier, the communication within the home had to be omitted. However, the work done on the proof-of-concept solution portrayed in section 3.4.6 is still a valid solution for how the communication between the user terminal and the AMS server could be accomplished.

The suggested layout presented in Figure 43 was created to fulfill the requirements given by WT, but there are enhancements to be made also here if the control modules support it. Instead of only being able to turn appliances on/off manually in the application, in some cases more detailed options would be better:

- For electrical radiators and other heating equipment the on/off button could be replaced with a select button letting the user choose a preferred temperature.
- Allowing triggers to be set on the individual control modules so they are activated at chosen times. Eventually setting time intervals for when an appliance should be turned on and off.
 - If the user owns an electric vehicle, it would be very beneficial to only recharge when the price is at its lowest every day. For example the user can specify how long it takes for the vehicle to recharge, as well as specifying the time it needs to be fully

charged. Then the system will automatically activate charging at the most profitable time according to the hourly prices shown in the pricelist.

- A timer could be set on all the heating appliances in the house, allowing the user to state a time period for which heating is not needed. Examples are during the night and when there's no one in the house. If the user states that he will be sleeping from 23:00 until 07:00, the temperature in the house could be turned down at 23:00 and then turned back on at 06:00 so the house will be at the user's preferred temperature when he gets out of bed. This will save electricity without interfering with the welfare of the household. An illustration of how this could look like is shown in Figure 91.

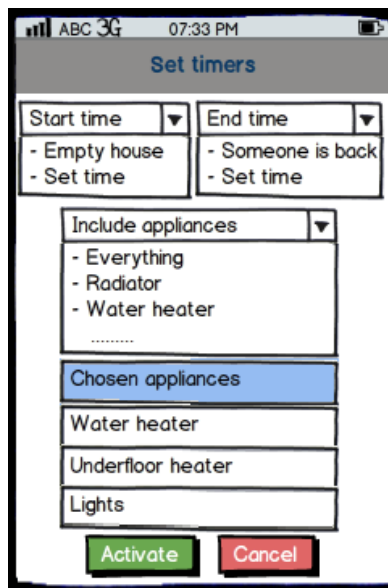


Figure 91: A timer functionality could help save electricity

- The tracker functionality mentioned in section 5.4 could also come in handy in this part of the application. By adding this functionality, chosen appliances could automatically be turned off when the house is empty. It could also be combined with the timer functionality shown in Figure 91 by adding an *Empty house* and a *Someone is back* option. This

would allow the chosen appliances be turned off when the house is empty, but still the house could be at a preferred temperature by the time the user returns home. An option to always turn off all lights and reduce the power to the heating system in the house while the house is empty could also be a valuable feature.

In addition to all these control, an option to add and remove appliances to the control list would be a useful enhancement. To minimize the efforts required by the users, this should be realized as an adaptive solution. By installing a new control module in the house it should automatically detect the control unit and be handed an internal address in the local network. It would then simply require entering the control module's id number in the application for authentication by the server. The server would then send a request to the control unit, asking for confirmation that a new control module with the given id number is present, and then add the new control module to the user's *control_module* table in the database, shown in Figure 64.

5.6. SMS Notifications

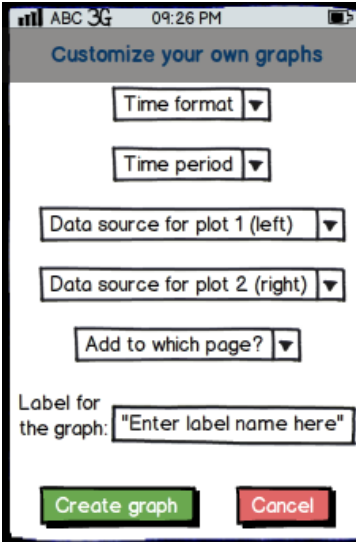
In the initial requirements presented by WT, given in Appendix B – Initial requirements given by WT, was an SMS-notification feature for alerting users about dramatically increased prices. This feature was considered not to be important for the pilot project, and was therefore not prioritized at this stage. However, it is a desirable feature for the commercial version of the application, and there are different ways of realizing it:

- Letting the user specify a price limit for what the user considers to be alarming. For this option to be a valuable feature, measures need to be taken to prevent spamming of messages.
 - A single SMS could be sent out every day at a chosen time, listing every hour the next day where the price is over the given limit.
- Algorithms could also be created to calculate what should be considered to be a dramatically increase in the given price area
 - This option is more difficult to design, but the gain from it might be higher since most people probably don't have a conscious relationship to the hour-by-hour prices on electricity.
 - In the same way as mentioned above, an SMS could be sent when the prices for the next day is collected, presenting the results of the algorithm.

The user could also be presented with the option to get daily messages giving the entire pricelist for the next day. This would keep the user updated and more likely to perform the necessary actions to get noticeable results, and hence continue using the application.

5.7. User-Customized Graphs

In the current prototype the configuration of the graphs on the Start page and the History page are hardcoded and therefore identical for all users of the application. Since the JS function, shown in Figure 58, generating the graphs was designed to be generic (see section 3.4.5), the function can easily be reused in order to make a solution where the users can configure their own graphs. An extra button for *add graphs* in the Settings page could link to a new page where the user is presented with options to create a new graph, like the sketch in Figure 92.



The image shows a mobile application interface for customizing graphs. The screen is titled "Customize your own graphs". It features five dropdown menus: "Time format", "Time period", "Data source for plot 1 (left)", "Data source for plot 2 (right)", and "Add to which page?". Below these is a text input field labeled "Label for the graph:" with the placeholder text "Enter label name here". At the bottom, there are two buttons: a green "Create graph" button and a red "Cancel" button.

Figure 92: A proposal for a user-customized graph functionality

The aforementioned jQuery command `.append()` is suited for realizing this functionality since it simply appends the chosen code, in this case a new graph element, to the existing HTML code without changing anything to the pre-existing code.

5.8. Comparison with Similar Houses and Friends Feature

If this system grows to have a large customer base, other possibilities may present themselves. To give the users incentive to be more energy efficient, information about how other users in similar houses are doing could be presented in the application. The anonymity of this feature in the Smarties application is easily achieved by storing usage data from similar houses in categories leaving out all user-specific data.

A different feature in the other end of the scale would be to allow users to have “friends” within the system. By enabling users to get information about their friends’ progress, an element of competition could add an extra motivation for certain users to be extra energy-efficient. How much, and which information to share with friends must naturally be controlled by the users themselves to avoid unpleasant experiences. If this type of implementation would become a success, it would also allow for environmental groups to award the best users with prizes, giving even more motivation to be energy-efficient.

According to a case study performed by the IBM Corporation [59], the use of social networking in a system like this delivered good results where the participating households reduced their consumption by up to eleven percent.

5.9. Scaling the System

As the tests in section 3.5.2 have shown, the AMS web application requires server hardware with better CPU and memory specifications than the current server used in the pilot project. However, if the system is to be a nationwide AMS provider, different ways of improving the systems scaling properties must be considered. One such action is load balancing, as shown in Figure 93.

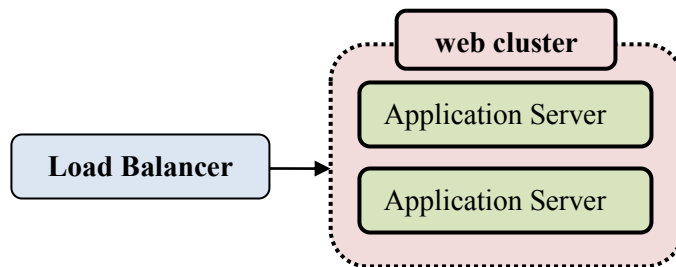


Figure 93: A load balancer can help improve the system’s scaling properties

By balancing the total load over multiple physical servers, the separate workload for each server can be reduced. As the amount of data increases, it could also be smart to consider partitioning the data in database clusters. Figure 94 illustrates an architecture where the mentioned methods are combined.

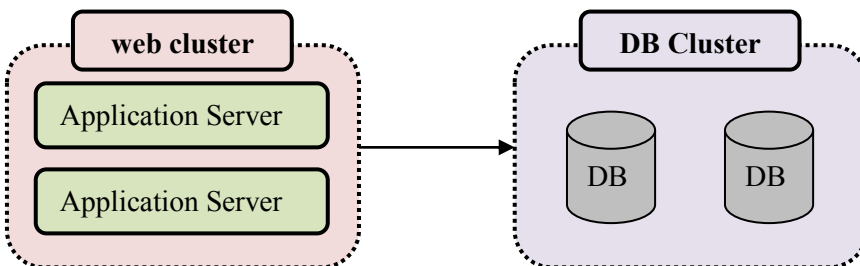


Figure 94: Creating database clusters is a way of scaling the application to an increased amount of data

6. Conclusion

In this thesis, an AMS system development study has been elaborated on. The focus has been on the web application development process where the different technology and design related choices have been reasoned. The work resulted in a functioning prototype which has gone through both performance- and security tests, and is now live on WT's servers.

Although rotations internally in the project group led to uncertainties regarding which home automation devices were to be used, a proof-of-concept solution has been developed for the device control functionality. By doing so, it has been proven that controlling devices in the home can be done in a secure and user-friendly way using the AMS web application.

As part of the critical assessment, shortcomings have been discovered and various value-adding features have been proposed for future releases. Among the shortcomings is an inconsistent process for meter-data retrieval which may cause sudden voids in the consumption data. It was very fortunate that this problem was found at an early stage since it must be corrected before the Smarties group can become a serious AMS service provider. However, a solution based on gathering readings from the smart meter logs after periods with missing data has been presented.

However, the prototype was successfully finalized on time and the Smarties system is currently serving multiple pilot customers. These users will provide valuable feedback on the usability and the usefulness of the system. In addition, this feedback will give an indication of whether or not the Smarties project has been successful in motivating customers to make changes to their consumption patterns.

It is believed that this system will succeed in raising awareness among users on how easy and profitable it is to be more energy efficient, thus a system like this will benefit the environment as well as helping people save money.

References

- [1] TechTerms.com, "The Tech Terms Computer Dictionary," [Online]. Available: <http://www.techterms.com/search>. [Accessed 18 Desember 2011].
- [2] Store Norske Leksikon, "EU 20-20-20," [Online]. Available: http://snl.no/EU_20-20-20. [Accessed 31 May 2012].
- [3] U.S. Department of Energy, "Centerpoint Energy and U.S. Deputy Secretary of Energy Daniel Poneman Announce Results of Pilot Project on Home Energy Use," [Online]. Available: http://apps1.eere.energy.gov/news/progress_alerts.cfm/pa_id=580. [Accessed 31 May 2012].
- [4] Wikipedia, "Smart meter," [Online]. Available: http://en.wikipedia.org/wiki/Smart_meter. [Accessed 02 May 2012].
- [5] B. Lawson and R. Shartp, *Introducing HTML5*, Berkeley, CA: Pearson Education, 2010.
- [6] N. Energinett, "Narvik Energinett," 2012. [Online]. Available: <http://www.narvikenerginett.no/en/Bedrift/AMS---Fjernmaling/>.
- [7] Hafslund ASA, "The power market in Norway - in brief," [Online]. Available: http://www.hafslund.no/english/customer_services/artikler/les_artikkel.asp?artikkelid=204. [Accessed 2 June 2012].
- [8] Leksikon, Store Norske, "Store Norske Leksikon," [Online]. Available:

- <http://snl.no/energiloven>. [Accessed 5 March 2012].
- [9] NASDAQ OMX, "Trade at NASDAQ OMX Commodities Europe's Financial Market," NASDAQ OMX, Oslo, 2011.
- [10] Nord Pool Spot, "About us," Nord Pool Spot, [Online]. Available: <http://www.nordpoolspot.com/About-us/>. [Accessed 6 March 2012].
- [11] R. Bræk og Ø. Haugen, Engineering Real Time Systems - An Object Oriented Methodology using SDL, Hemel Hempstead: Prentice Hall, 1993.
- [12] P. Vixie, "Cron - daemon to execute scheduled commands," [Online]. Available: <http://unixhelp.ed.ac.uk/CGI/man-cgi?cron+8>. [Accessed 15 May 2012].
- [13] Google, "Google.org - About Us," [Online]. Available: <http://www.google.org/about.html>. [Accessed 18 April 2012].
- [14] R. Miller, "Google's PowerMeter to let users track electricity usage," Engadget, 9 February 2009. [Online]. Available: <http://www.engadget.com/2009/02/09/nyt-googles-powermeter-to-let-users-track-electricity-usage/>. [Accessed 23 April 2012].
- [15] Google, "Google PowerMeter - Save Energy. Save Money. Make a Difference.," [Online]. Available: <http://www.google.com/powermeter/about/>. [Accessed 18 April 2012].
- [16] AlertMe, "AlertMe - Creating smart homes," [Online]. Available: <http://www.alertme.com/>. [Accessed 24 April 2012].
- [17] AlertMe, "AlertMe Platform," [Online]. Available: <http://www.alertme.com/business/platform.html>. [Accessed 26 April 2012].
- [18] AlertMe, "Demo on how to save energy & be energy efficiend - AlertMe," [Online]. Available: http://www.alertme.com/try_the_demo. [Accessed 23

April 2012].

- [19] T. Stevens, "Powering Google's PowerMeter: testing TED 5000 and AlertMe Energy," Engadget, 8 December 2009. [Online]. Available: <http://www.engadget.com/2009/12/08/powering-googles-powermeter-testing-ted-5000-and-alertme-energ/>. [Accessed 23 April 2012].
- [20] Energy Inc, "TED The Energy Detective," [Online]. Available: <http://www.theenergydetective.com/>. [Accessed 24 April 2012].
- [21] Energy Inc, "How TED works," [Online]. Available: <http://www.theenergydetective.com/about/howtedworks>. [Accessed 24 April 2012].
- [22] Energy Inc, "TED 5000 Series Installation Guide," [Online]. Available: <http://files.theenergydetective.com/QuickStartInstallation%20v110711.pdf>. [Accessed 25 April 2012].
- [23] Energy Inc, "TED 5000 FAQ," [Online]. Available: <http://www.theenergydetective.com/ted-5000/frequently-asked-questions>. [Accessed 25 April 2012].
- [24] J. Hunter og W. Crawford, Java Servlet Programming, Sebastopol: O'Reilly Media Inc, 2001.
- [25] Python Software Foundation, "About Python," [Online]. Available: <http://www.python.org/about/>. [Accessed 29 May 2012].
- [26] Python Software Foundation, "Glossary - Python v2.7.3 documentation," [Online]. Available: <http://docs.python.org/glossary>. [Accessed 29 May 2012].
- [27] Techearth.net, "Python:Basics:All about Python," [Online]. Available: http://techearth.net/python/index.php5?title=Python:Basics:All_about_Python#Python_Compilation_Process. [Accessed May 30 2012].

- [28] Python Software Foundation, "WebFrameworks," [Online]. Available: <http://wiki.python.org/moin/WebFrameworks>. [Accessed 30 May 2012].
- [29] Django Software Foundation, "Django - The Web framework for perfectionists with deadlines," [Online]. Available: <https://www.djangoproject.com/>. [Accessed 30 May 2012].
- [30] J. Kaplan-Moss and A. Holovaty, *The Definitive Guide to Django: Web Development Done Right*, Berkeley: Apress, 2008.
- [31] C. McDonough, *The Pyramid Web Application Development Framework: Version 1.0*, Agendaless Consulting, 2011.
- [32] S. W. Davies, "Curia - The Great Web Framework Shootout," [Online]. Available: <http://blog.curiasolutions.com/the-great-web-framework-shootout/>. [Accessed May 30 2012].
- [33] K. Sobti, "A Look at Some JavaScript Alternatives," 11 December 2011. [Online]. Available: http://devworks.thinkdigit.com/Features/A-Look-at-Some-JavaScript-Alternatives_8131.html. [Accessed 14 May 2012].
- [34] Adobe Systems Incorporated, "Adobe Flash Platform," [Online]. Available: <http://www.adobe.com/flashplatform/>. [Accessed 14 May 2012].
- [35] Wikipedia, "ActionScript," [Online]. Available: <http://en.wikipedia.org/wiki/ActionScript>. [Accessed 14 May 2012].
- [36] Sencha Inc., "Sencha Touch 2 Build Mobile Web Apps with HTML5," [Online]. Available: <http://www.sencha.com/products/touch/>. [Accessed 14 May 2012].
- [37] E. Spencer, "Ext JS 4 Preview: Faster, Easier, More Stable," [Online]. Available: <http://www.sencha.com/blog/ext-js-4-preview-faster-easier-more-stable>. [Accessed 14 May 2012].

- [38] Refsnes Data, "Browser Statistics," [Online]. Available: http://www.w3schools.com/browsers/browsers_stats.asp. [Accessed 14 May 2012].
- [39] jQuery Project, "History - jQuery Project," [Online]. Available: <http://jquery.org/history>. [Accessed 27 April 2012].
- [40] The jQuery Foundation, "Announcing jQuery Mobile 1.0," [Online]. Available: <http://jquerymobile.com/blog/2011/11/16/announcing-jquery-mobile-1-0/>. [Accessed 30 April 2012].
- [41] The jQuery Mobile Project, "About jQuery Mobile," [Online]. Available: <http://jquerymobile.com/demos/1.1.0-rc.1/docs/about/>. [Accessed 30 April 2012].
- [42] KNX Association, "KNX," 17 June 2010. [Online]. Available: <http://www.knx.org/knx/what-is-knx/>. [Accessed 27 February 2011].
- [43] ZigBee Alliance, "ZigBee Technology," [Online]. Available: <http://www.zigbee.org/About/AboutTechnology/ZigBeeTechnology.aspx>. [Accessed 7 May 2012].
- [44] ZigBee Alliance, "ZigBee Specification Overview," [Online]. Available: <http://www.zigbee.org/Specifications/ZigBee/Overview.aspx>. [Accessed 7 May 2012].
- [45] Z-Wave Alliance, "About Z-Wave," [Online]. Available: <http://www.z-wave.com/modules/AboutZ-Wave/>. [Accessed 7 May 2012].
- [46] Z-Wave Alliance, "Z-Wave.com | FAQ," [Online]. Available: http://www.z-wave.com/modules/xoopsfaq/index.php?cat_id=2#q17. [Accessed 7 May 2012].
- [47] Shifra Smart Homes, "Smart Home Automation Technologies in the UAE," [Online]. Available: <http://www.shifrasmarthomes.com/AR001.html>.

- [Accessed 7 May 2012].
- [48] IETF, "The IP Network Address Translator (NAT)," 1994. [Online]. Available: <http://www.ietf.org/rfc/rfc1631.txt>. [Accessed 27 May 2012].
- [49] IETF, "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)," 2008. [Online]. Available: <http://tools.ietf.org/html/rfc5128#section-3.3>. [Accessed 31 May 2012].
- [50] The jQuery Foundation, "Scripting pages," [Online]. Available: <http://jquerymobile.com/demos/1.1.0/docs/pages/page-scripting.html>. [Accessed 20 May 2012].
- [51] O. Laursen, "flot - Attractive Javascript plotting for jQuery," [Online]. Available: <http://code.google.com/p/flot/>. [Accessed 26 May 2012].
- [52] The Arduino Team, "Arduino," [Online]. Available: <http://www.arduino.cc/>. [Accessed 22 May 2012].
- [53] M. Schmidt, "Meet the Arduino," [Online]. Available: <http://pragprog.com/magazines/2010-07/meet-the-arduino>. [Accessed 22 May 2012].
- [54] Google Developers, "Chrome Developer Tools: Overview," [Online]. Available: <https://developers.google.com/chrome-developer-tools/docs/overview>. [Accessed 31 May 2012].
- [55] M. Pilgrim, HTML5: Up and Running - Dive into the Future of Web Development, O'Reilly Media, 2010.
- [56] Microsoft Corporation, "Preserving Session State in Network Load Balancing Web Server Clusters (IIS 6.0)," [Online]. Available: <http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/77cb4318-75f8-4310-a05f-3605b5768007.msp?mfr=true>. [Accessed 29 May 2012].

- [57] SmartBear, "LoadComplete - Web Load Testing," [Online]. Available: <http://smartbear.com/products/qa-tools/load-testing-tool>. [Accessed 10 April 2012].
- [58] OWASP Foundation, OWASP Testing Guide V3.0, OWASP Foundation, 2008.
- [59] IBM Corporation, "City of Dubuque - Investing in sustainability for future generations and future prosperity," [Online]. Available: <http://www.ibm.com/smarterplanet/us/en/leadership/dubuque/assets/pdf/Dubuque.pdf>. [Accessed 30 May 2012].
- [60] *SMARTIES - Proposal template*, 2011.
- [61] EMCC, "The European Market," 2012. [Online]. Available: <http://www.marketcoupling.com/market-coupling/european-market>. [Accessed 5 March 2012].
- [62] Oracle, "Essentials, Part 1, Lesson 1: Compiling & Running a Simple Program," [Online]. Available: <http://www.oracle.com/technetwork/java/compile-136656.html#platform>. [Accessed 29 May 2012].
- [63] A. Das, H. Pung, F. Lee and L. Wong, "Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet," in *IFIP*, Singapore, 2008.
- [64] NASDAQ OMX Group, Inc, "About N2EX," 2011. [Online]. Available: <http://www.n2ex.com/aboutn2ex/>. [Accessed 6 March 2012].

Appendix A – The Smarties proposal template



SmartIES – Proposal template

Deadline to submit a proposal: 01.10.2011

- 1. Chosen Future Scenario (from the report which scenario are you proposing a solution to)?**

The proposed solution has technology to support both scenarios, but the main focus would be on scenario 2.

- 2. Participant presentation**

Partner 1	SIEMENS
Country	International – main contact in Norway
Contact person	Arne Petter Kjøraas
Contact email	arne.kjoeraas@siemens.com
Contact phone	+47 922 53 285
Website	Siemens.com

Partner 2	Wireless Trondheim
Country	Norway
Contact person	Kjartan Mjøsund
Contact email	kjartan@tradlosetrondheim.no
Contact phone	+47 926 17 408
Website	www.tradlosetrondheim.no

Partner 3	Kamstrup
Country	Denmark
Contact person	Gustaf Troeng
Contact email	GUT@kamstrup.com
Contact phone	455 12 644
Website	http://www.kamstrup.dk

Partner 4	Trønder Energi
Country	Norway
Contact person	Cathrine Tronstad
Contact mail	Cathrine.Tronstad@tronderenergi.no
Contact phone	+47 900 27 080
Website	www.tronderenergi.no

Partner 5	Energy Company
Country	in Sweden, Denmark or Iceland
Contact person	Two energy Companies in Sweden and Denmark have been approached with positive feedback (without commitments) from Luleå Energy.

3. Description of the proposed solution

The proposed solution is a demonstration concept with 3 goals

- to show how easy it is to save energy using new technology
- test a new technology for AMS potentially reducing the cost of AMS with 30-50 % utilizing existing infrastructure
- show how the end user and the society can benefit from the massive rollout of AMS planned throughout Europe by 2020

The concept consists of three main “modules”:

- i) Wi-Fi as transport for AMS
- ii) A SIEMENS product called “Synco Living”
- iii) An app showing among other things real-time energy consumption

80 % of all households in Norway has Wi-Fi installed already, and the rate is expected to be similar in most of Europe within a few years. This makes using Wi-Fi as the AMS-traffic carrier attractive, from both a deployment/installation point of view, and an economical point of view. As of now, the main technologies as transport for AMS are GSM/3G, Power Line communication (PLC) and radio mesh network (requires deployment of an additional, dedicated infrastructure). GSM/3G and especially PLC have many drawbacks including capacity of the canal. Building a dedicated radio mesh infrastructure is extremely costly. By utilizing the Wi-Fi already

present in households it is expected that the lifetime costs for communication to AMS can be reduced with 30-50 %.

Synco-living enables the user to remote monitor and regulate the power in her/his household. With Synco-Living one can switch on and off electrical devices in the home in a way that leads to lower energy consumption and better comfort.

The web-app part of the concept will have as a basic functionality to show real-time energy consumption on the user's smart phone and on the web.

The proposed project is a demonstration project, and throughout the project the end users will be followed up for the purpose of looking for changes in consumption, awareness etc. At the end of the project the results will be gathered in a brochure. The brochure is meant to be a teaser for others by showing how easy it is to save energy/ how little effort it takes to achieve it, and to present the possibilities of controlling the home with Synco-Living. The brochure will be a "how-to" for end users including costs of installation and a short product overview, and it will highlight the benefits and what possibilities this gives for the end user. Want to be a smart energy user? Just do like this!

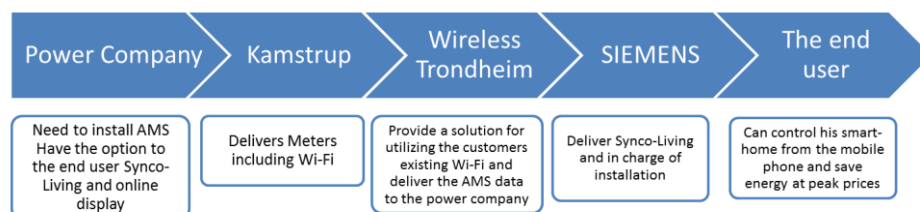
4. Description of the Business model

- a. Value chain description (how will you get the product to the market?):

The plan is to market and sell the concept to power companies when they ask for tenders for AMS roll-out. Synco-Living, with the possibilities to monitor and switch on and off boiler, heater cables, radiator and lightening will be offered to the power companies to make it possible for the power companies to offer these solutions as an option to the end user when rolling out AMS. In the AMS-rollout, an electrician has to visit the end user to install a new meter. To install Synco-Living, little if any additional work has to be done by the electrician; that is, a package that enables the end user to monitor and control his usage can be almost fully, and easily, installed by the end user her-/himself. In this way, the end user can get an affordable solution to control his smart home and save power

and money. Also, in this way the end user will feel and appreciate the usefulness/value of AMS.

The value chain will be as follows:



The value proposition for each of the parties is:

Power company

- Significantly reduced costs of installing and running AMS (30-50 %)
- The carrier (Wi-Fi) can be used also for other value added services, providing an opportunity for the power company to provide more value and to make more profit
- Offer the customer something extra when installing AMS – the customer will gain something from the rollout. She/he is not forced to have a new meter without any visible benefits.

Kamstrup

- Beat the competitors by offering a more cost efficient life time solution and thereby deliver more meters

Wireless Trondheim

- Deliver a solution for using existing Wi-Fi to the power companies. Getting paid by the power company.
- Offer an interface for providing the AMS data to the end user, offered for free or as a paid service (with Synco-Living)

SIEMENS

- An efficient sales channel for Synco-Living
- Make more money on installation

The end user

- Getting a smart home at a low cost (far less than ordering a smart home by himself) any readily installed together with AMS without any hassle or knowledge needed.
- The possibility to save energy using his smart home

5. How will the users be involved

- a. What is the value in participating for the users:
- b. How many users are needed for the pre-pilot (approx.)?
- c. How long will the pre-pilot last (approx.)?
- d. How many users are needed for the cross-border pilot (approx.)?
- e. How long will the cross-border pilot last (approx.)?

The aim of the project is to make a proof-of-concept and to get feedback from the end-users to be able to determine whether they find the concept valuable or not, as well as how they use the concept, what they want more of etc. To be able to do this we need a qualitative study of selected households.

For the pre-pilot we will use 2 installations. The pre-pilots will be installed at the end of November 2011. There is no plan to end the pre-pilot. The pre-pilot is planned to run till the end of the project with different modifications and testing. However, the consortium is prepared to make a project report showing the results of the pre-pilot at the beginning of January 2012 (meaning a report for the first month).

The pre-pilot will include an AMS over Wi-Fi reading, a full Synco-Living installation (though with limited functionality enabled) and a first version of an online display and control application for monitoring of the power consumption and controlling of the different devices.

For the cross-border pilot we will do an installation of 8-10 households in 2 countries, 4-5 households per country. At least 2 apartments and at least 2 single family houses will be chosen for each country.

The cross border pilot will be installed in April 2012 and last till June 2012. The reason we want the pilot to be installed in April is to get at least 2 full months for feedback and monitoring of the end user. If possible, we would like the cross border pilot to last over the summer holidays to see if the end users use the application controlling their home when they are on vacation.

The chosen households will either get the installation for free or for a small charge. The reason to have a cost is to recruit end users that are motivated; that is, users that are likely to spend some effort using their smart home and to provide us with valuable feedback. Also, we believe that finding users willing to pay will ensure that these users represent the market that the concept will meet when commercialized. We would like input from the Smarties projects on the approach.

6. Describe

- a. The value of the end product for future users
- b. Contribution to innovation (How will your proposed end product or services contribute to innovation?):
- c. Benefit to the society (How will your proposed end product or services benefit the society?)

The user will be offered smart home functionality when AMS is installed in her/his house. The smart house functionality is offered to a significantly lower price than if the user orders the product her-/himself. Also, packaging AMS with the smart house product this way makes installation (of the latter product) completely hassle-free for the user.

This is due to the fact that an electrician will have to visit the user to install a meter anyway, and installing the smart house product requires only marginal additional effort from the electrician. In other words, two installations in one visit both eliminate hassle for the end user and reduce cost. The cost for the end user is further reduced due to a combination of i) the volume discount attained

by a power company that orders a large amount of smart house packages and ii) the value of the product-bundling and lock-in effect that the power company gets with the offering (meaning that the power company can justify for itself lower margins on the smart house product). With the smart home the end user will be able to control his home and to save energy through different measures. The aim is that the concept will be attractive to order because the end user sees the benefits. With AMS over Wi-Fi the end user is able to control certain devices in the house and see the effects immediately online using web or his smart phone.

The project contributes to innovation in several ways. First, the rollout costs of AMS could be reduced dramatically using existing infrastructure (Wi-Fi). Compared to other technologies the life-time costs could be reduced with 30-50 % (or much more compared to building a mesh radio network only for AMS). In Norway, the cost of the AMS rollout is expected to exceed 10 Billion NOK, half of it being related to the transportation canal. The potential is therefore huge. AMS using Wi-Fi is not deployed in Europe and Kamstrup is one of few meter vendors developing a module that enables their meter to use Wi-Fi. Kamstrup's meters with Wi-Fi are not ready for the market yet, but is in a development stage that makes it possible to use in a development project, and the development project is expected to take the meters to market ready state.

Neither Synco-Living nor anonline display for web or smart phones are technical innovations. Both solutions exist, but so far the volume of smart-houses using these solutions is low. This is due to the fact that the end user needs very specific knowledge to order this and the fact that the smart home functionality is dependent on AMS not yet installed. The packaging of technology and offer it together with the installation of AMS makes it easy for the user to choose a smart home solution, and the solution will be provided to an affordable price. Because smart house technology will be easy to order and because it will be offered at an affordable price, it is expected that a large percentage of the households will choose the concept. This will in turn reduce the total energy consumption.

The concept will have major impact of innovation in 2 parts;

1. The business model enabling smart home technology for the end user in a very cost efficient way
2. Using Wi-Fi for AMS – using Wi-Fi has potential huge savings for the lifetime of the installation. There are no European companies delivering meters with Wi-Fi ready for commercial use as of today (there are American companies, but they use Wi-Fi for AMR not for AMS). Kamstrup have been developing a Wi-Fi module for their meters but this hasn't been tried out in a large scale with customers like in this project. This project will be a proof-of-concept projects for Kamstrup.

Both the potential savings using Wi-Fi for AMS and the reduced power consumption will benefit society and the environment. If all expected gains can be realized, the savings for society is huge; as an example, in Norway the savings could be 2-3 Billion NOK for the rollout and costs of AMS. Benefits for the society using Synco-Living would be smarter homes and thereby the possibility to save energy, which will have a positive effect on the environment. This way Synco-Living will contribute to increase the effect of AMS and thereby reducing the peaks and levelling the power consumption. This will in turn lead to reduced need to build new power lines saving both the environment and the society for large costs.

It is also considered a positive effect that the users see that they will get a benefit from AMS. In many countries there is a debate on the fact that AMS by itself will not give any effects for the end users. It is expected that the end users will be negative to taking the cost of AMS while getting no gains. With the proposed concept it is easy to show the benefits for the end user.

7. Objectives

The projects objectives are:

- To show how easy it is to save energy using new technology
- Test a new technology for AMS potentially reducing the cost of AMS with 30-50 % utilizing existing infrastructure

- Show how the end user and the society can benefit from the massive rollout of AMS planned throughout Europe by 2020

8. Milestones and/or Gantt chart

Tasks	Responsible	2011		2012								
		Oct	Nov	Dec	Jan	Febr	Mar	Apr	May	Jun	Jul	
Delivering of meters with Wi-Fi module	Kamstrup	20.okt										
Developing of AMS Wi-Fi proof of concept	Wireless T		M1					M2				
Developing of online display	Wireless T		M1					M2				
Programming of Synco-Living	SIEMENS		M1					M2				
Determining end-user for installing pre-pilot	Trønder Energi		20.nov									
Preparing to get meter readings	Trønder Energi											
Installing of AMS and synco-living for pre pilot	SIEMENS		30.nov									
Gather user experience pre-pilot	Wireless T											
Reporting to SmartIES on pre-pilot	Wireless T				10.jan							
Determining end-user for installing cross-border pilot	Energy Company											
Preparing to get meter readings	Energy Company											
Installing of AMS and synco-living for cross-border pilot	SIEMENS							M2				
Gather user experience cross-border pilot	SmartIES partner											
Design of brochure	SIEMENS											
Reporting to SmartIES	SIEMENS/Wireless Trondheim											

The projects GANTT will be shifted 1 month, with start around 20th of November instead of 20th of October.

9. Task description

The different tasks and milestones are presented in the GANTT chart in chapter 8.

Due to the tight time schedule working on the proposal we have not decided upon, or made agreements with a power company in either Sweden, Denmark or Iceland. Contact approaches have been made and it has been clarified that SIEMENS can deliver and install in both Denmark and Sweden (most likely also in Iceland, but we have not been talking to the Iceland office yet). Luelå Energy (Sweden) have stated a positive interest given that the project will be realized, but have not made any commitments. An energy company in Denmark has also been approached. Therefor during 2011 it is an important task will be to recruit and select a power company from either Sweden, Denmark or Iceland.

10. Describe how you plan to collaborate cross border

- a. What partners are involved and what are their roles?
- b. How do you plan to test the solution cross border (set up etc.)?
- c. Others role in the proposed solution (SmartIES partners, etc.)?

The different partners and their roles are presented in chapter 4 “value chain”.

The concept is planned tested in 2 countries (could be extended to more if a larger budget is allocated).

SIEMENS will be in charge of installation of both AMS and Synco-Living in both countries. They will include the respectively departments in the country of the pilot. SIEMENS will have a project manager coordinating and working on the different sites. SIEMENS will also be in charge of producing the brochure at the end of the project.

Wireless Trondheim will be in charge of developing the AMS over Wi-Fi solution (this is a project Wireless Trondheim have been working on the last year), development of online display solution and in charge of gathering user experience and reporting of this to SIEMENS.

Kamstrup will be in charge of delivering AMS meters with Wi-Fi prepared module and working together with, and offering support to Wireless Trondheim and SIEMENS.

The energy companies will be partners collecting AMS data and act as learning partners on how to utilize AMS and Synco-Living.

The project will be dependent upon that SmartIES partners from the selected country can contribute in gathering user experience.

II. Budget / Cost estimates

The SmartIES project will max fund 50% of salary cost – Travel cost will not be paid by the SmartIES project.

- a. Your contribution
- b. Others contribution
- c. Expected fund by SmartIES
- d. Expected fund by others

The budget is given on next page. Traveling costs will be covered by the companies. Traveling cost and time for participation on the final conference has been added to the previous version. The financial support from SmartIES will be used to equipment and time for developing, installing and documenting/reporting. Each participant will contribute with about 50 % own effort so that the support is in percentage spread approximately even.

For installation local companies in the areas will be used supervised by SIEMENS. These costs are not shown directly in the budget but the are included in other budget posts.

An additional comment; Kamstrup indirectly receive more funding from the project than listed in the budget. The reason for this is that Wireless Trondheim will buy equipment from Kamstrup that is listed in Wireless Trondheim's budget. This is done so because Wireless Trondheim can reuse this equipment later and it would make more sense if Wireless Trondheim own this equipment after the projects.

Attachment: Additional information about the partners

SIEMENS is a global company present In 192 countries. This means that SIEMENS has a global distribution system for bringing new products to market. SIEMENS have developed Synco-Living a "low cost" solution for smart homes with the aim to offer affordable smart home technology to the mass market.

Kamstrup is one of the largest vendors of power meters in Europe and are present in most European countries, Russia and in the Middle East. Kamstrup is one of the few vendors in Europe developing Wi-Fi as an alternative transport module to GSM/3G, PLC and mesh radio.

Wireless Trondheim is a company within wireless (Wi-Fi) and wireless value added services. Wireless Trondheim realized a citywide wireless network in 2006 making Trondheim one of Europe first wireless cities. In 2011 Wireless Trondheim realized the world's largest service for indoor navigation (still in beta). Wireless Trondheim is a leading company within the niche of wireless value added services over Wi-Fi.

The partners all have complementing knowledge, services and products and all have a direct motivation to go-to market with their solutions. This increase the likelihood of successfully launching a new concept to the market.

Appendix B – Initial requirements given by WT

Functional requirements:

1. The application shall have a start page showing information the user can act on
 - a. Real time usage in Kilowatt hours (KWh) and Norwegian Kroner (NOK)
 - b. Electricity used today and yesterday in KWh and NOK
 - c. Show the highest- and average price of the next 24 hours
 - d. Link to an area where the user can turn on/off electric equipment in the home
 - e. Link to historic data
 - f. Link to a settings page
2. There shall be a page/tab with historic data
 - a. A graph with many different options
 - i. Usage in KWh and NOK for the last year, month by month
 - ii. Usage in KWh and NOK for the last year, month by month correlated with temperature
 - iii. Usage over the last month on a day-to-day basis in KWh and NOK
3. There shall be a page/tab where the user can control devices in his home
 - a. All the devices in the list shall be able to turn on and off
 - b. The user shall be able to control the inside temperature
4. There shall be a settings page/tab
 - a. The user shall have the possibility of manually setting his position, or using GPS if possible, in order to get the correct price-data.
 - b. SMS-notification when prices increase dramatically
5. The application shall be designed in such a fashion that it in the future can collect data about electricity-prices from different providers, and from these data calculate the user's electricity bill for the last year/quarter.

Non-functional requirements:

1. Security

Non-authorized personnel shall not be able to control equipment in a user's home, nor get a hold of usage data.

Appendix C – Email correspondence with AlertMe

From: support@alertme.com
To: oystewaa@stud.ntnu.no
Date: Thu, 26 Apr 2012 05:50:36 -0400
Subject: Ticket #5474-10220932 Issue Resolved

Hello,

We're pleased to let you know that your issue has been resolved. Please review the resolution below and let us know if we can be of any further assistance regarding this issue.

Hi Oystein,

I am afraid our system would not work on 3-phase, or outside the UK.

The AlertMe system would not talk to your SmartMeter, but neither would it interfere with it - so if you had one the system would still be able to work.

The Reader fits around the live wire on the meter, rather than having the power pass through it, so the unit is not completely accurate, but should be at least 95% accurate.

Kind regards

Dominic

Appendix D – Appcache

With appcache activated, the browser checks if the manifest-file has been updated since last time it entered the webpage.

```
Application Cache Checking event
Application Cache Downloading event
Application Cache Progress event (0 of 22)
Application Cache Progress event (1 of 22)
Application Cache Progress event (2 of 22)
Application Cache Progress event (3 of 22)
Application Cache Progress event (4 of 22)
Application Cache Progress event (5 of 22)
Application Cache Progress event (6 of 22)
Application Cache Progress event (7 of 22)
Application Cache Progress event (8 of 22)
Application Cache Progress event (9 of 22)
Application Cache Progress event (10 of 22)
Application Cache Progress event (11 of 22)
Application Cache Progress event (12 of 22)
Application Cache Progress event (13 of 22)
Application Cache Progress event (14 of 22)
Application Cache Progress event (15 of 22)
Application Cache Progress event (16 of 22)
Application Cache Progress event (17 of 22)
Application Cache Progress event (18 of 22)
Application Cache Progress event (19 of 22)
Application Cache Progress event (20 of 22)
Application Cache Progress event (21 of 22)
Application Cache Progress event (22 of 22)
Application Cache UpdateReady event
```

Figure 95: All the files in the manifest file are downloaded when there has been an update since last time the page was loaded

```
Application Cache Checking event
Application Cache NoUpdate event
```

Figure 96: If no update has been done, the files are loaded from cache

Appendix











	Name Path	Size Content	Time Laten
1	 /oystewaa/smartyes/application/s /oystewaa/smartyes/application/smarti	14.16KB 13.80KB	32ms 27ms
2	 jquery-1.7.1.min.js /oystewaa/smartyes/application/smarti	92.04KB 91.67KB	19ms 7ms
3	 jquery.flot.min.js /oystewaa/smartyes/application/smarti	37.05KB 36.68KB	62ms 21ms
4	 jquery.flot.threshold.min.js /oystewaa/smartyes/application/smarti	1.26KB 910B	61ms 18ms
5	 jquery.mobile-1.0.1.min.css /oystewaa/smartyes/application/smarti	48.97KB 48.61KB	60ms 20ms
6	 dataFetchAndDraw.min.js /oystewaa/smartyes/application/smarti	32.91KB 32.54KB	60ms 20ms
7	 red-btn-theme.min.css /oystewaa/smartyes/application/smarti	9.77KB 9.41KB	59ms 19ms
8	 jquery.mobile-1.0.1.min.js /oystewaa/smartyes/application/smarti	82.08KB 81.70KB	80ms 28ms
9	 jquery.ui.datepicker.mobile.min.c /oystewaa/smartyes/application/smarti	1.75KB 1.40KB	59ms 19ms
10	 jQuery.ui.datepicker.min.js /oystewaa/smartyes/application/smarti	38.30KB 37.93KB	78ms 25ms
11	 jquery.ui.datepicker.mobile.min.j /oystewaa/smartyes/application/smarti	1.58KB 1.21KB	78ms 26ms
12	 styles.min.css /oystewaa/smartyes/application/smarti	4.15KB 3.79KB	8ms 3ms

Figure 97: Loading times without Apache













	Name Path	Size Content	Time Latency
	 /oystewaa/smartyes/application/s /oystewaa/smartyes/application/smarti	(from cache)	2ms 2ms
	 jquery.flot.min.js /oystewaa/smartyes/application/smarti	(from cache)	15ms 8ms
	 jquery.flot.threshold.min.js /oystewaa/smartyes/application/smarti	(from cache)	8ms 8ms
	 jquery-1.7.1.min.js /oystewaa/smartyes/application/smarti	(from cache)	8ms 7ms
	 dataFetchAndDraw.min.js /oystewaa/smartyes/application/smarti	(from cache)	15ms 11ms
	 jquery.mobile-1.0.1.min.css /oystewaa/smartyes/application/smarti	(from cache)	15ms 12ms
	 red-btn-theme.min.css /oystewaa/smartyes/application/smarti	(from cache)	14ms 12ms
	 jquery.ui.datepicker.mobile.min.c /oystewaa/smartyes/application/smarti	(from cache)	14ms 12ms
	 jquery.mobile-1.0.1.min.js /oystewaa/smartyes/application/smarti	(from cache)	14ms 11ms
	 jQuery.ui.datepicker.min.js /oystewaa/smartyes/application/smarti	(from cache)	15ms 12ms
	 jquery.ui.datepicker.mobile.min.j /oystewaa/smartyes/application/smarti	(from cache)	14ms 12ms
	 styles.min.css /oystewaa/smartyes/application/smarti	(from cache)	1ms 1ms

Figure 98: Loading times with Apache